(v. 2.2) Developer Guide

(v. 2.2) Getting started

In this section, you will learn how to:

- prepare development environment
- create new project
- launch demo
- configure and launch bot
- and many more

TIP
Useful materials and technologies used to development and run EasyRPA projects:

- Lombok - java library that converts annotations to java code
- Maven - project management and comprehension tool
- Git - distributed version control system
- Docker - platform for developers and sysadmins to build, run, and share applications with containers
- IntelliJ IDEA - IDE written in Java for developing computer software
- Selenium - web-browser automation tool

(v. 2.2) Development Environment Setup

Install JDK

As Java project, EasyRPA required JDK installed. Currently we are supporting minimal java version 8. We recommend to use Oracle JDK 8. Ensure `JAVA_HOME` environment variable is set and points to your JDK after install.

Install Maven

EasyRPA uses Maven for project building. You can download Apache Maven from official site. Extract distribution archive in any directory and add the unpacked distribution’s bin directory to your user `PATH` environment variable. Refer to Installing Apache Maven documentation for details.

Install Git

Git is great tool to version control. You can get installer for you OS here.

Then you will need to configure your local Git installation to use your GitHub credentials. Start command line and run commands:

```
git config --global user.name “github_username”
```

```
git config --global user.email “email_address”
```

Replace **github_username** and **email_address** with your GitHub credentials.

The **IDEA** environment contains Git plugin by default and you can use this feature for RPA development instead of downloading Git from site.

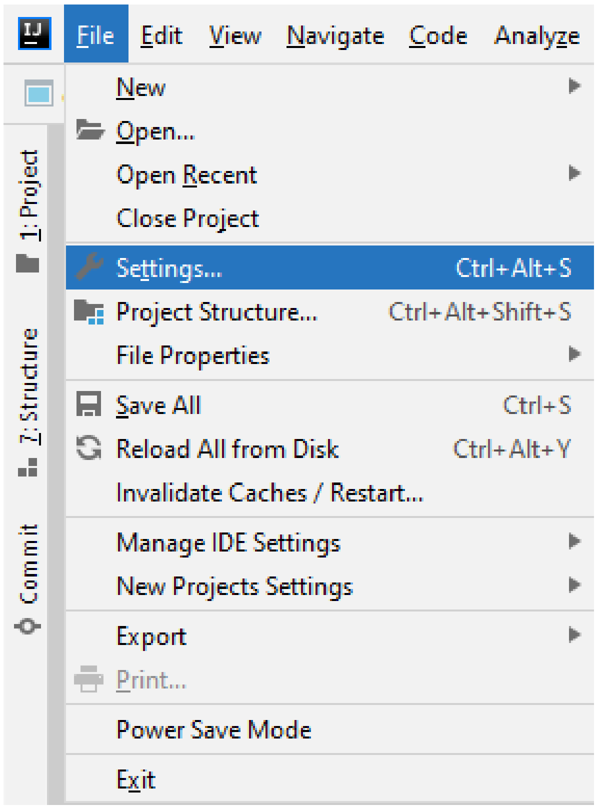
Install IDE

Though it's possible to work with EasyRPA in other IDEs, we recommend starting your acquaintance with IntelliJ IDEA Community Edition IDE. Get installer here.

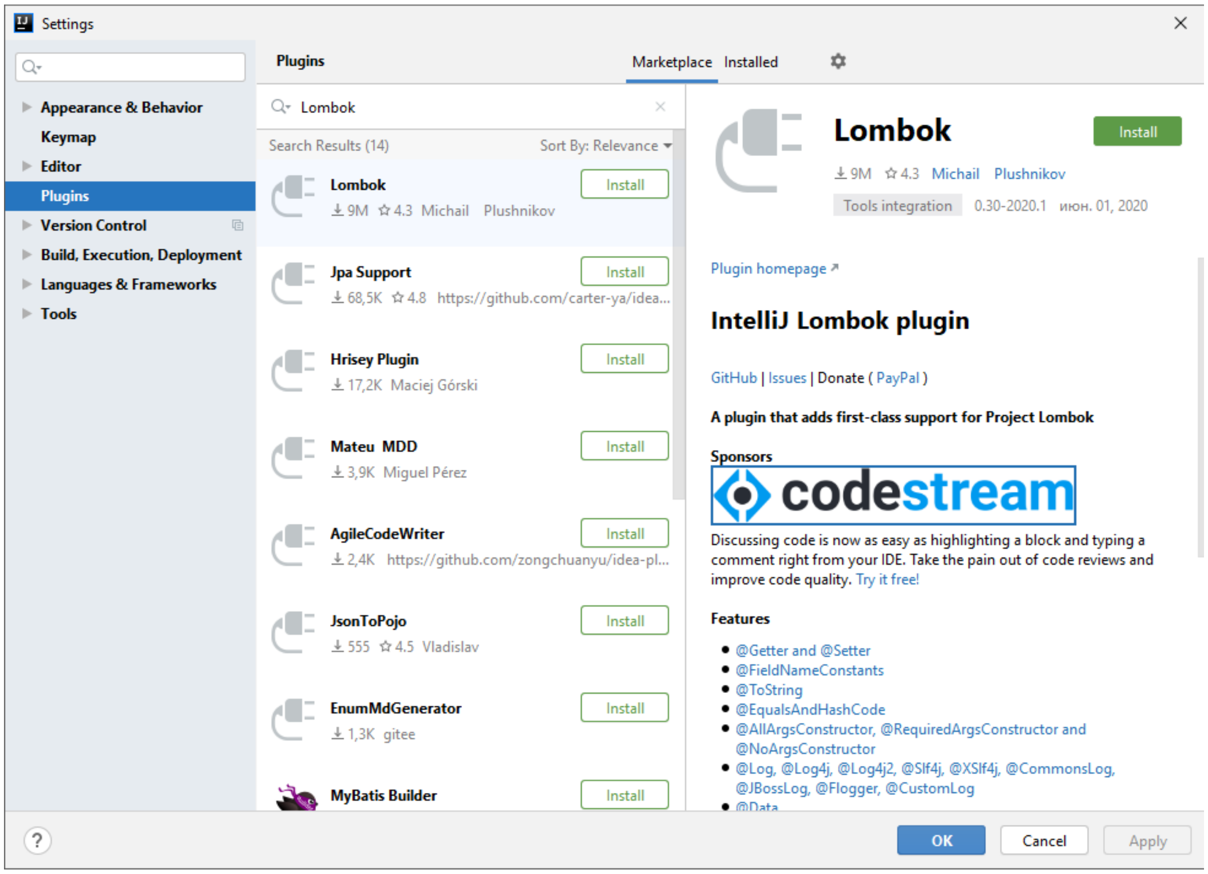
Install Lombok Plugin

Lombok plugin for IDEA provides more convenience in writing your code. You can check all its features on the official site.

Start IntelliJ IDEA IDE. Go to File Settings menu:



Select Plugins on the left panel, type *lombok* in search field and press *Enter*.



IntelliJ Lombok plugin

GitHub | Issues | Donate (PayPal)

A plugin that adds first-class support for Project Lombok

Sponsors

codestream

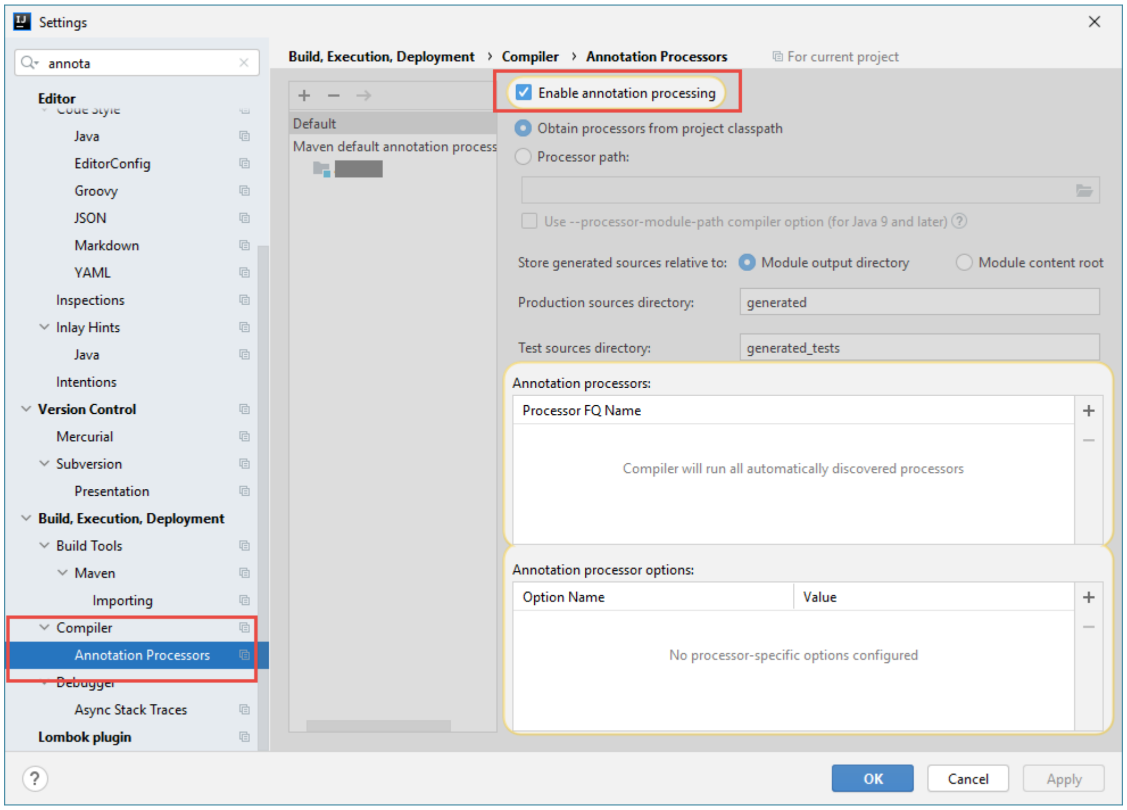
Discussing code is now as easy as highlighting a block and typing a comment right from your IDE. Take the pain out of code reviews and improve code quality. Try it free!

Features

- @Getter and @Setter
- @FieldNameConstants
- @ToString
- @EqualsAndHashCode
- @AllArgsConstructor, @RequiredArgsConstructor and @NoArgsConstructor
- @Log, @Log4j, @Log4j2, @Slf4j, @XSlf4j, @CommonsLog, @JBossLog, @Flogger, @CustomLog
- @Data

Click *Install* button for *Lombok* and restart IDE.

Switch annotation processing on:



[✓] Enable annotation processing

[● Obtain processors from project classpath]

Processor path:

Use --processor-module-path compiler option (for Java 9 and later)

Store generated sources relative to: [● Module output directory] Module content root

Production sources directory: generated

Test sources directory: generated_tests

Annotation processors:

Processor FQ Name

Compiler will run all automatically discovered processors

Annotation processor options:

Option Name	Value

No processor-specific options configured

Add Certificate to Truststore

1. Import the certificate to the truststore

```
<JAVA_HOME>/java/bin/keytool -keystore <JAVA_HOME>/java/jre/lib/security/cacerts -import -file
easyRpaCertificate.crt -alias easyRpa
```

2. You will be asked to enter keystore password.
   Java: The initial password of the "cacerts" keystore file is "changeit".
3. If the key already exists and/or it's not correct, so you need to delete that **alias**:

```
<JAVA_HOME>/java/bin/keytool -delete -alias easyRpa -keystore <JAVA_HOME>/java/jre/lib/security
/cacerts
```

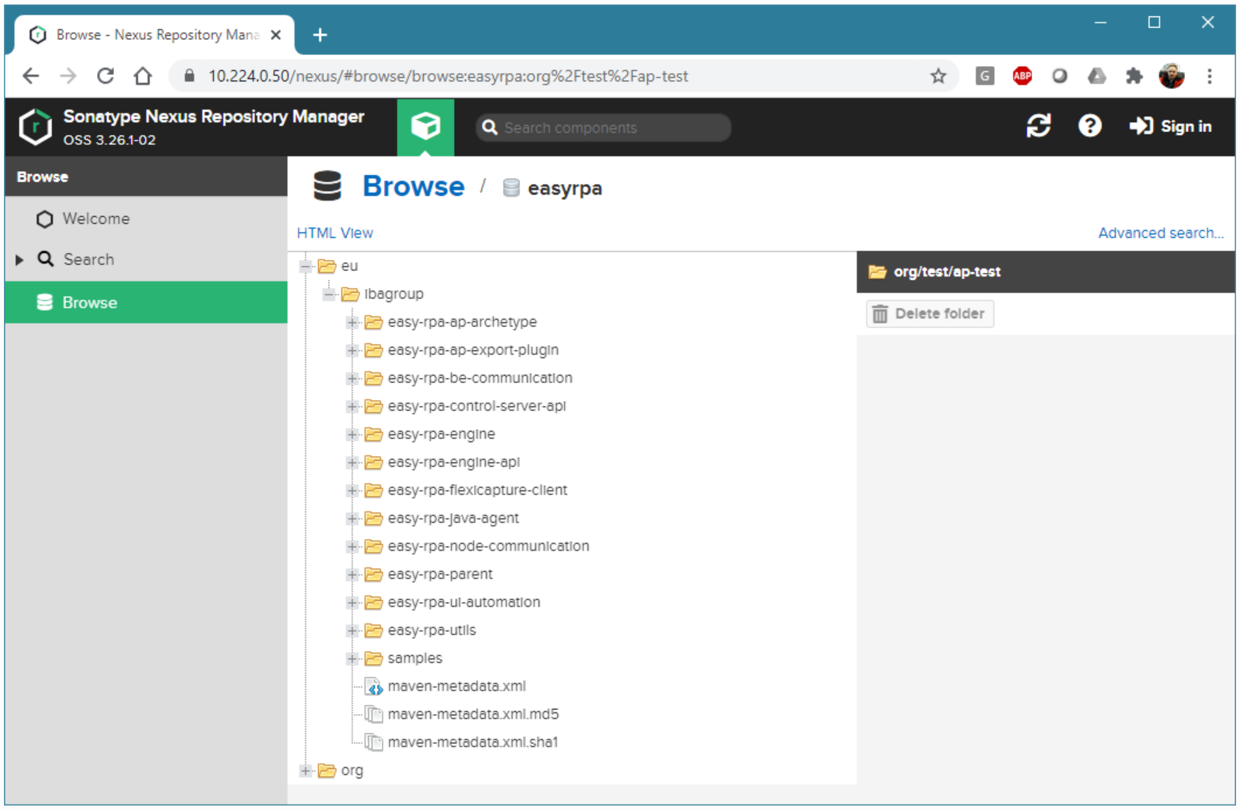
(v. 2.2) Quick Start Guide

EasyRPA Control Server instance

All stuff you need to use during development process are coming together with Control Server installation in onboard nexus:

Nexus URL

https://<cs url>/nexus/#browse/browse:easyrpa:org%2Ftest%2Fap-test



Refer to the installation guide to obtain development CS instance or use an existing to which you have an access.

Setup maven to use EasyRPA nexus repositoryproject

Open your maven **settings.xml** (under ~/.m2 on linux or C:/Users/<your user>/.m2 on windows), and add maven repository from the provided templates:

For server maven authentication if you are planning to deploy your Automation Process jar code on CS's nexus:

```
        <servers>

                . . . .
                <server>
                        <id>easy-rpa-nexus</id>
                        <username><CS nexus deployment user></username>
                        <password><CS deployment password></password>
                </server>

        </servers>
```

Control Server maven repository for artifacts access:

```
        . . . . .
        <profiles>
                <profile>
                        <id>easyrpa</id>
                        <repositories>
                                <repository>
                                        <id>easy-rpa-nexus</id>
                                        <url>https://<CS UTR>/nexus/repository/easyrpa/</url>
                                </repository>
                        </repositories>
                        <pluginRepositories>
                                <pluginRepository>
                                        <id>easy-rpa-nexus</id>
                                                <url>https://<CS UTR>/nexus/repository/easyrpa/</url>
                                </pluginRepository>
                        </pluginRepositories>
                </profile>
        </profiles>
        <activeProfiles>
                <activeProfile>easyrpa</activeProfile>
        </activeProfiles>

        . . . .
```

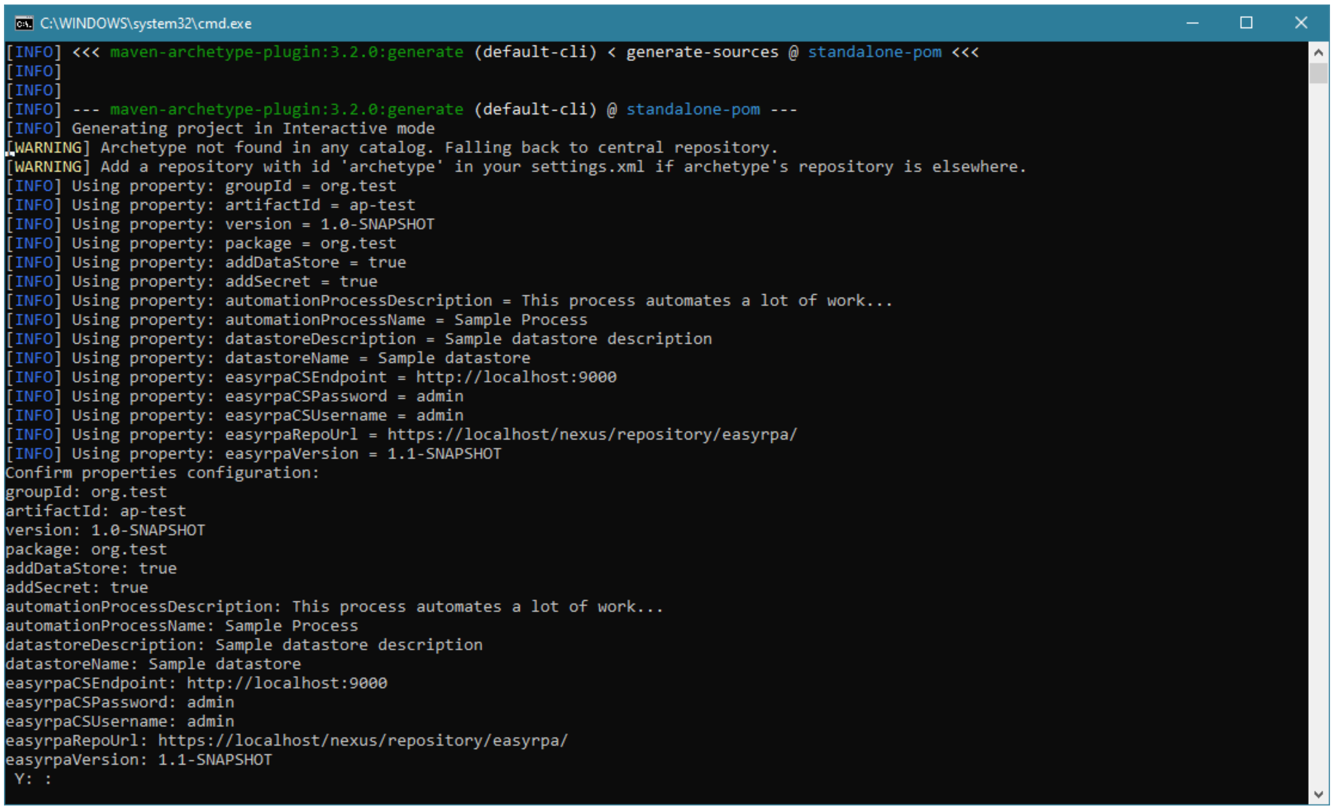
Generate project from archetype

We are going to use maven **easy-rpa-ap-archetype** to create a new project from a template. Use the following command to run wizard:

```
mvn archetype:generate -DarchetypeGroupId=eu.ibagroup -DarchetypeArtifactId=easy-rpa-ap-archetype -
DarchetypeVersion=<easyRPA version> -DgroupId=<your project group id> -DartifactId=<your project artifact
id> -Dversion=<your project version> -DaddSecret=true
```

Make sure that you added Control Server root certificate into your java trust store.

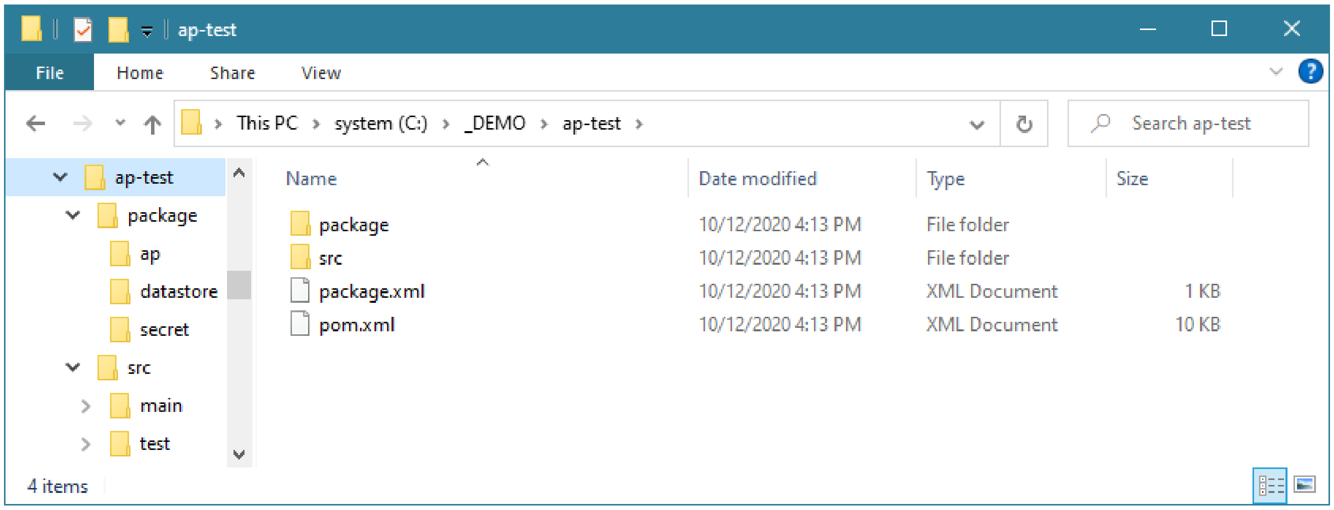
The archetype generator will provide you pre-filed values for project generation:



```
[INFO] <<< maven-archetype-plugin:3.2.0:generate (default-cli) < generate-sources @ standalone-pom <<<
[INFO]
[INFO]
[INFO] --- maven-archetype-plugin:3.2.0:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Interactive mode
[WARNING] Archetype not found in any catalog. Falling back to central repository.
[WARNING] Add a repository with id 'archetype' in your settings.xml if archetype's repository is elsewhere.
[INFO] Using property: groupId = org.test
[INFO] Using property: artifactId = ap-test
[INFO] Using property: version = 1.0-SNAPSHOT
[INFO] Using property: package = org.test
[INFO] Using property: addDataStore = true
[INFO] Using property: addSecret = true
[INFO] Using property: automationProcessDescription = This process automates a lot of work...
[INFO] Using property: automationProcessName = Sample Process
[INFO] Using property: datastoreDescription = Sample datastore description
[INFO] Using property: datastoreName = Sample datastore
[INFO] Using property: easyrpaCSEndpoint = http://localhost:9000
[INFO] Using property: easyrpaCSPassword = admin
[INFO] Using property: easyrpaCSUsername = admin
[INFO] Using property: easyrpaRepoUrl = https://localhost/nexus/repository/easyrpa/
[INFO] Using property: easyrpaVersion = 1.1-SNAPSHOT
Confirm properties configuration:
groupId: org.test
artifactId: ap-test
version: 1.0-SNAPSHOT
package: org.test
addDataStore: true
addSecret: true
automationProcessDescription: This process automates a lot of work...
automationProcessName: Sample Process
datastoreDescription: Sample datastore description
datastoreName: Sample datastore
easyrpaCSEndpoint: http://localhost:9000
easyrpaCSPassword: admin
easyrpaCSUsername: admin
easyrpaRepoUrl: https://localhost/nexus/repository/easyrpa/
easyrpaVersion: 1.1-SNAPSHOT
 Y: :
```

You can use the default or say N and run the wizard in interactive mode.

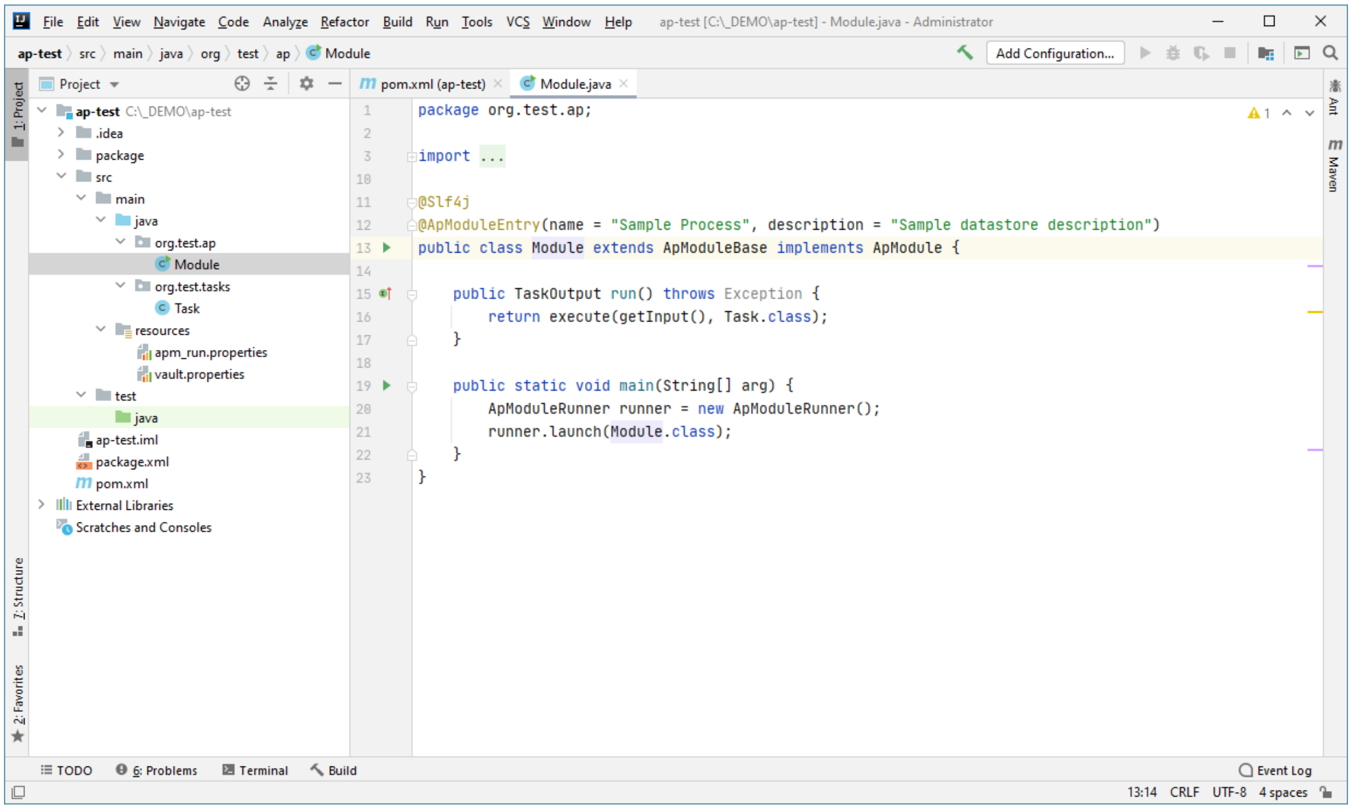
The result will look as following:



Name	Date modified	Type	Size
package	10/12/2020 4:13 PM	File folder	
src	10/12/2020 4:13 PM	File folder	
package.xml	10/12/2020 4:13 PM	XML Document	1 KB
pom.xml	10/12/2020 4:13 PM	XML Document	10 KB

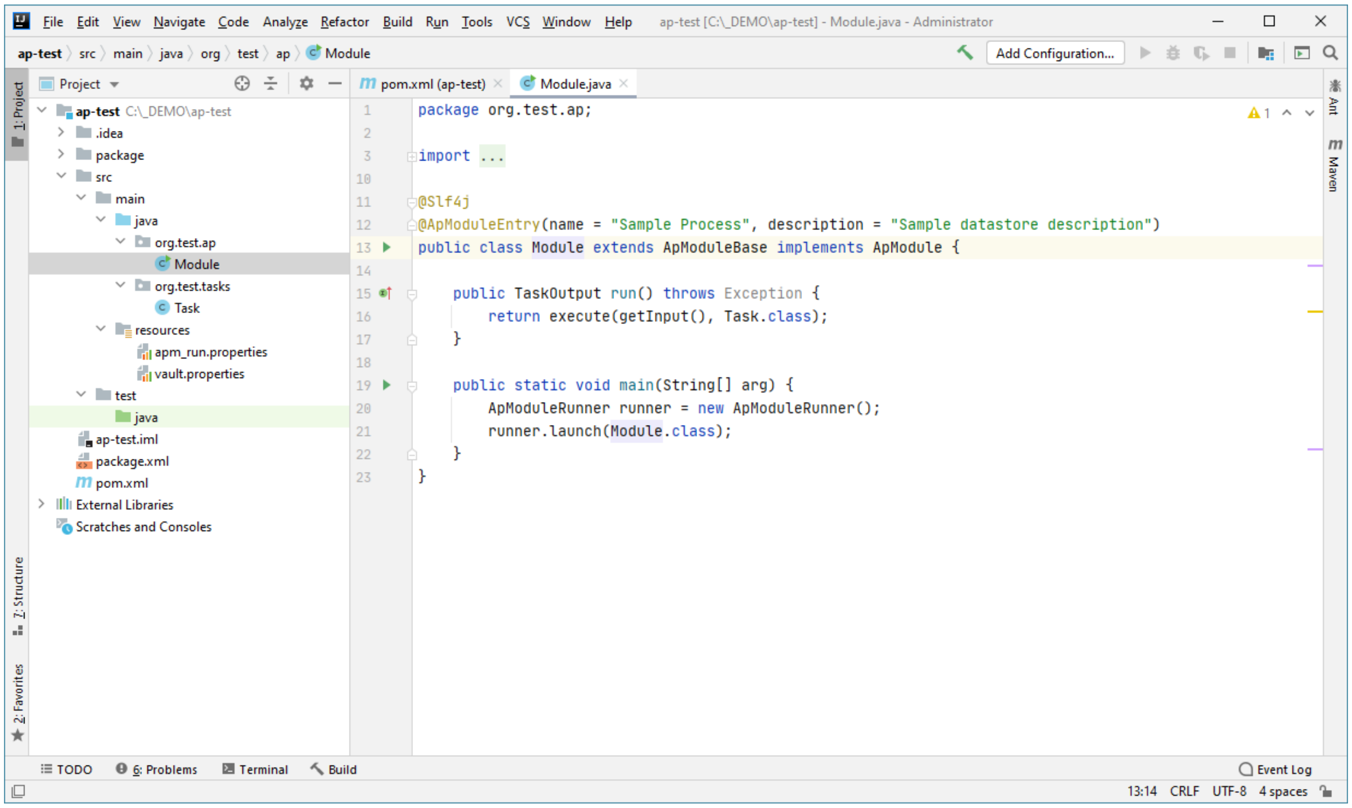
Build and run locally

Open the generated project in IDEA:



```java
package org.test.ap;

import ...

@Slf4j
@ApModuleEntry(name = "Sample Process", description = "Sample datastore description")
public class Module extends ApModuleBase implements ApModule {

    public TaskOutput run() throws Exception {
        return execute(getInput(), Task.class);
    }

    public static void main(String[] arg) {
        ApModuleRunner runner = new ApModuleRunner();
        runner.launch(Module.class);
    }
}
```

Find the <Your Module> class and run it:

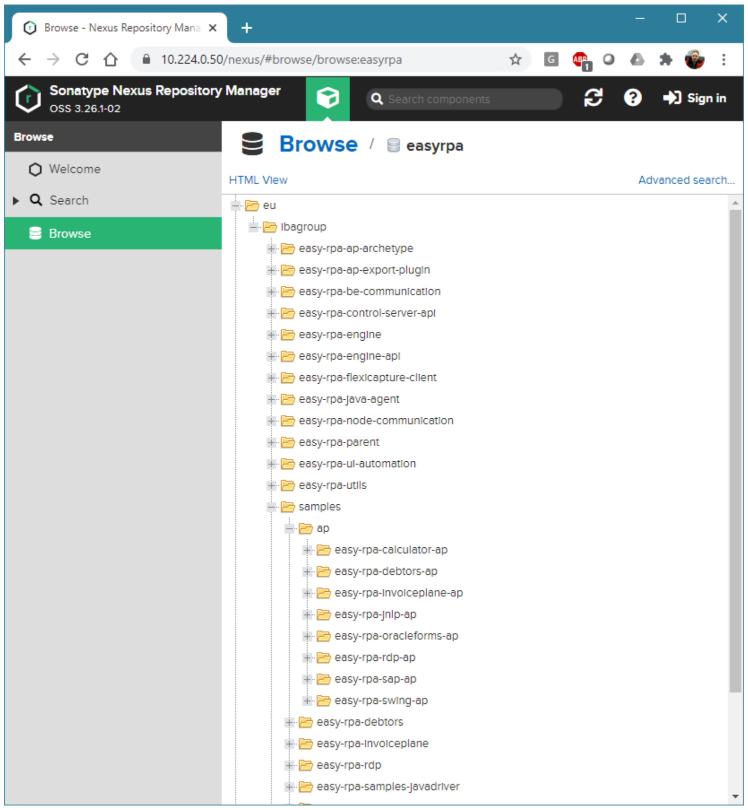


```java
package org.test.ap;

import ...

@Slf4j
@ApModuleEntry(name = "Sample Process", description = "Sample datastore description")
public class Module extends ApModuleBase implements ApModule {

    public TaskOutput run() throws Exception {
        return execute(getInput(), Task.class);
    }

    public static void main(String[] arg) {
        ApModuleRunner runner = new ApModuleRunner();
        runner.launch(Module.class);
    }
}
```

The generated task class do not have any useful code. Let's add something you can play with.

Adding RPA code

Default setup of easyRPA nexus contains sample modules and codes you can play with:



```
eu
  ibagroup
    easy-rpa-ap-archetype
    easy-rpa-ap-export-plugin
    easy-rpa-be-communication
    easy-rpa-control-server-api
    easy-rpa-engine
    easy-rpa-engine-api
    easy-rpa-flexicapture-client
    easy-rpa-java-agent
    easy-rpa-node-communication
    easy-rpa-parent
    easy-rpa-ui-automation
    easy-rpa-utils
    samples
      ap
        easy-rpa-calculator-ap
        easy-rpa-debtors-ap
        easy-rpa-invoiceplane-ap
        easy-rpa-jnlp-ap
        easy-rpa-oracleforms-ap
        easy-rpa-rdp-ap
        easy-rpa-sap-ap
        easy-rpa-swing-ap
      easy-rpa-debtors
      easy-rpa-invoiceplane
      easy-rpa-rdp
      easy-rpa-samples-javadriver
```

Let's use sample tasks classes in our code to make a quick RPA module.

Add the following dependency into your project's **pom.xml**:

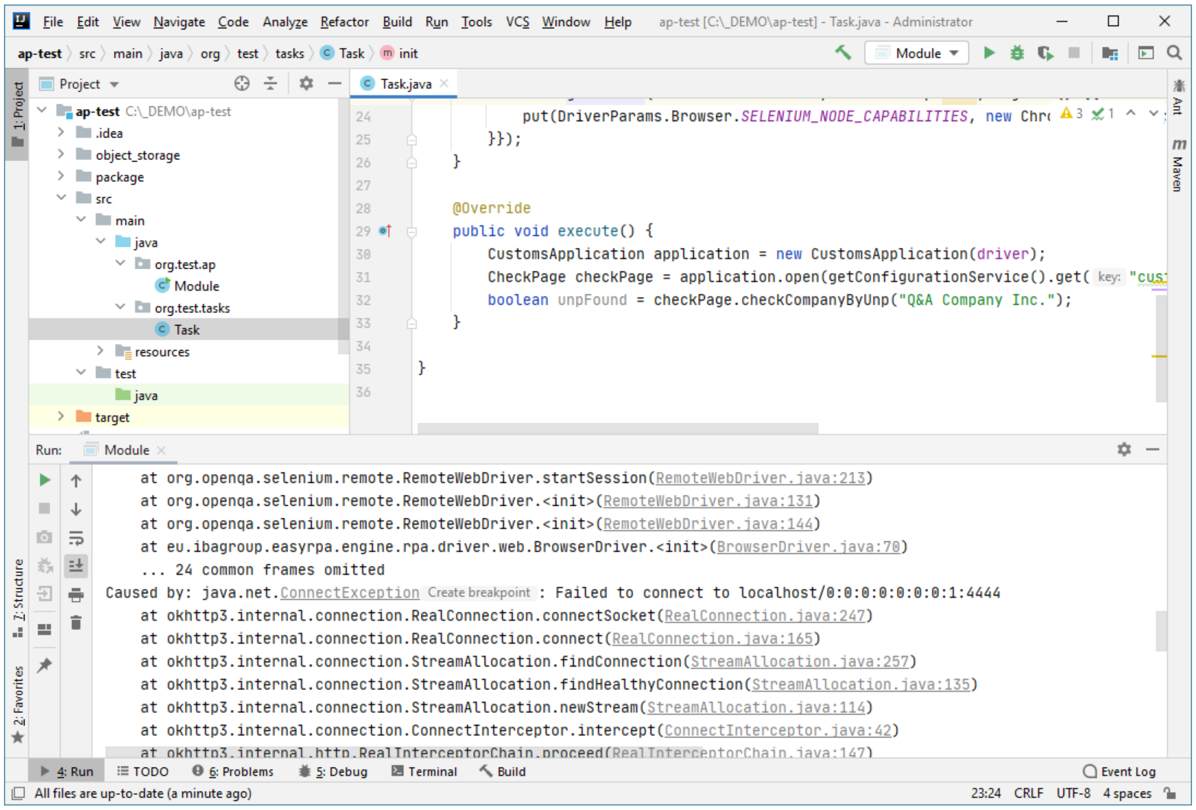
```
        <dependency>
            <groupId>eu.ibagroup.samples.system</groupId>
            <artifactId>easy-rpa-customs-system</artifactId>
            <version>1.1.0</version>
        </dependency>
```

Click on pom.xml and reload the project.

Update your module with calling of RPA task:

```
package org.test.tasks;

import eu.ibagroup.easyrpa.engine.annotation.AfterInit;
import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApTaskBase;
import eu.ibagroup.easyrpa.engine.rpa.driver.DriverParams;
import eu.ibagroup.easyrpa.engine.rpa.driver.web.BrowserDriver;
import eu.ibagroup.easyrpa.system.customs.CustomsApplication;
import eu.ibagroup.easyrpa.system.customs.page.CheckPage;
import lombok.extern.slf4j.Slf4j;
import org.openqa.selenium.chrome.ChromeOptions;

import java.util.HashMap;

@Slf4j
@ApTaskEntry(name = "Sample RPA Task", description = "EasyRPA quick Task")
public class Task extends ApTaskBase {

    private BrowserDriver driver;

    @AfterInit
    public void init() {
        driver = getDriver(BrowserDriver.class, new HashMap<Enum, Object>() {{
            put(DriverParams.Browser.SELENIUM_NODE_CAPABILITIES, new ChromeOptions());
        }});
    }

    @Override
    public void execute() {
        CustomsApplication application = new CustomsApplication(driver);
        CheckPage checkPage = application.open(getConfigurationService().get("customes.url", "http://www.gtk.
gov.by/ru/dolg-test-ru/"));
        boolean unpFound = checkPage.checkCompanyByUnp("Q&A Company Inc.");
    }

}
```

Run the task, it will fail with Selenium hub connection url:



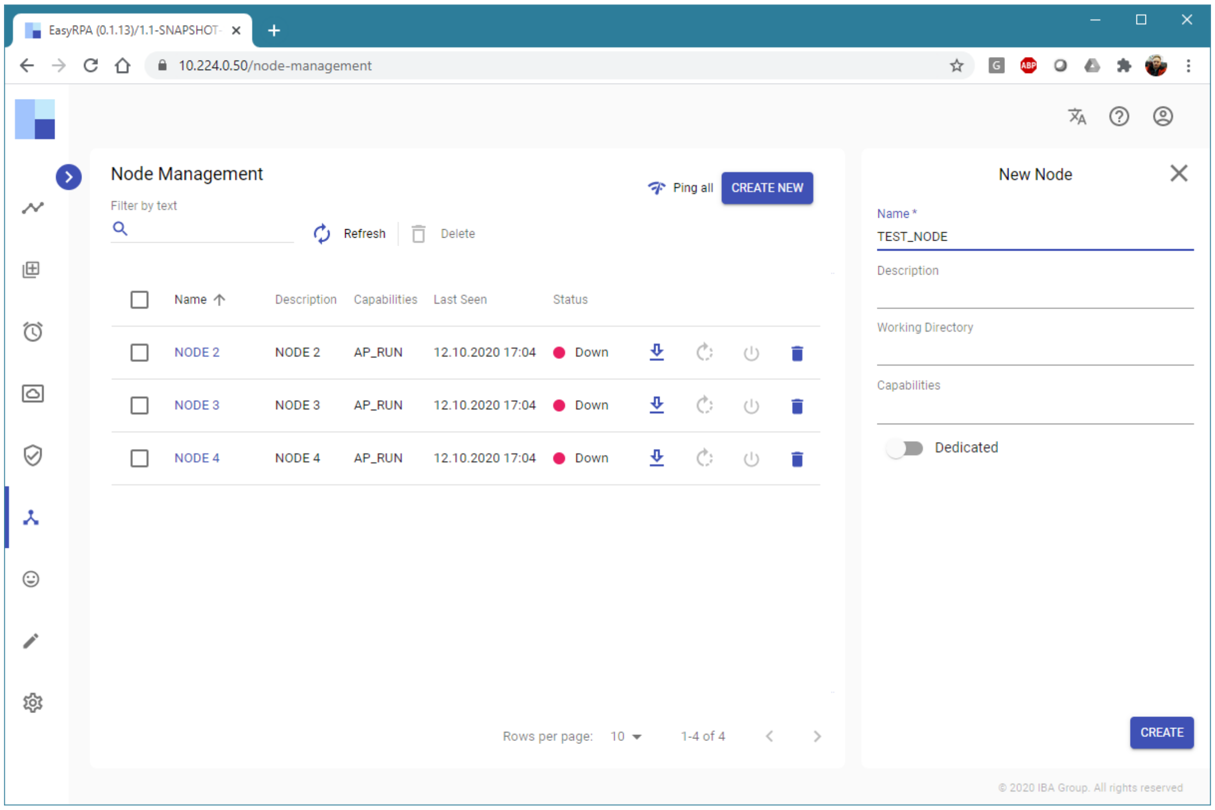
```
            put(DriverParams.Browser.SELENIUM_NODE_CAPABILITIES, new Chr
        }});
    }

    @Override
    public void execute() {
        CustomsApplication application = new CustomsApplication(driver);
        CheckPage checkPage = application.open(getConfigurationService().get( key: "cus
        boolean unpFound = checkPage.checkCompanyByUnp("Q&A Company Inc.");
    }

}
```

```
    at org.openqa.selenium.remote.RemoteWebDriver.startSession(RemoteWebDriver.java:213)
    at org.openqa.selenium.remote.RemoteWebDriver.<init>(RemoteWebDriver.java:131)
    at org.openqa.selenium.remote.RemoteWebDriver.<init>(RemoteWebDriver.java:144)
    at eu.ibagroup.easyrpa.engine.rpa.driver.web.BrowserDriver.<init>(BrowserDriver.java:70)
    ... 24 common frames omitted
Caused by: java.net.ConnectException Create breakpoint : Failed to connect to localhost/0:0:0:0:0:0:0:1:4444
    at okhttp3.internal.connection.RealConnection.connectSocket(RealConnection.java:247)
    at okhttp3.internal.connection.RealConnection.connect(RealConnection.java:165)
    at okhttp3.internal.connection.StreamAllocation.findConnection(StreamAllocation.java:257)
    at okhttp3.internal.connection.StreamAllocation.findHealthyConnection(StreamAllocation.java:135)
    at okhttp3.internal.connection.StreamAllocation.newStream(StreamAllocation.java:114)
    at okhttp3.internal.connection.ConnectInterceptor.intercept(ConnectInterceptor.java:42)
    at okhttp3.internal.http.RealInterceptorChain.proceed(RealInterceptorChain.java:147)
```

Setup node

You can setup Selenium server using selenium documentation, but the easiest way is to install and setup easyRPA node agent.

Go to control server nodes page and create a node for your machine:



Name	Description	Capabilities	Last Seen	Status
NODE 2	NODE 2	AP_RUN	12.10.2020 17:04	Down
NODE 3	NODE 3	AP_RUN	12.10.2020 17:04	Down
NODE 4	NODE 4	AP_RUN	12.10.2020 17:04	Down

New Node

Name * TEST_NODE

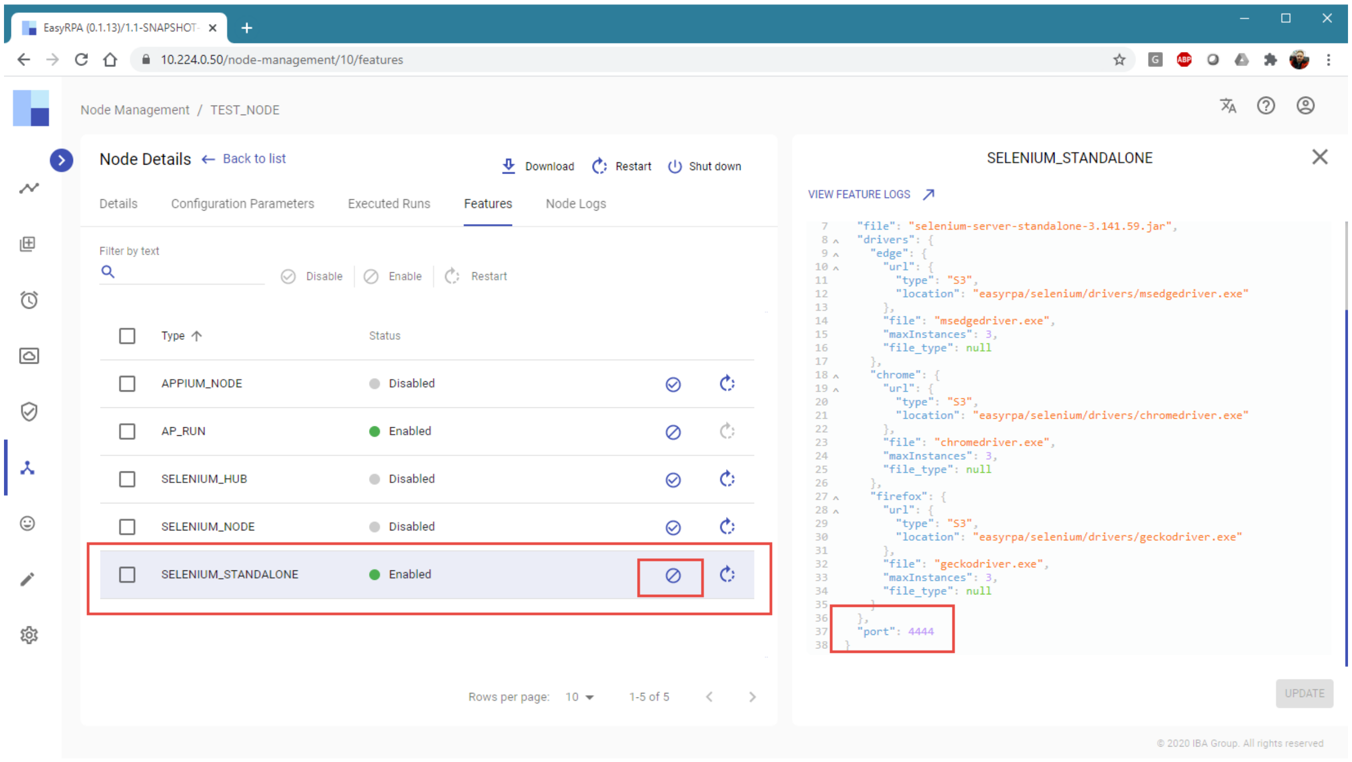
Description

Working Directory

Capabilities

Dedicated

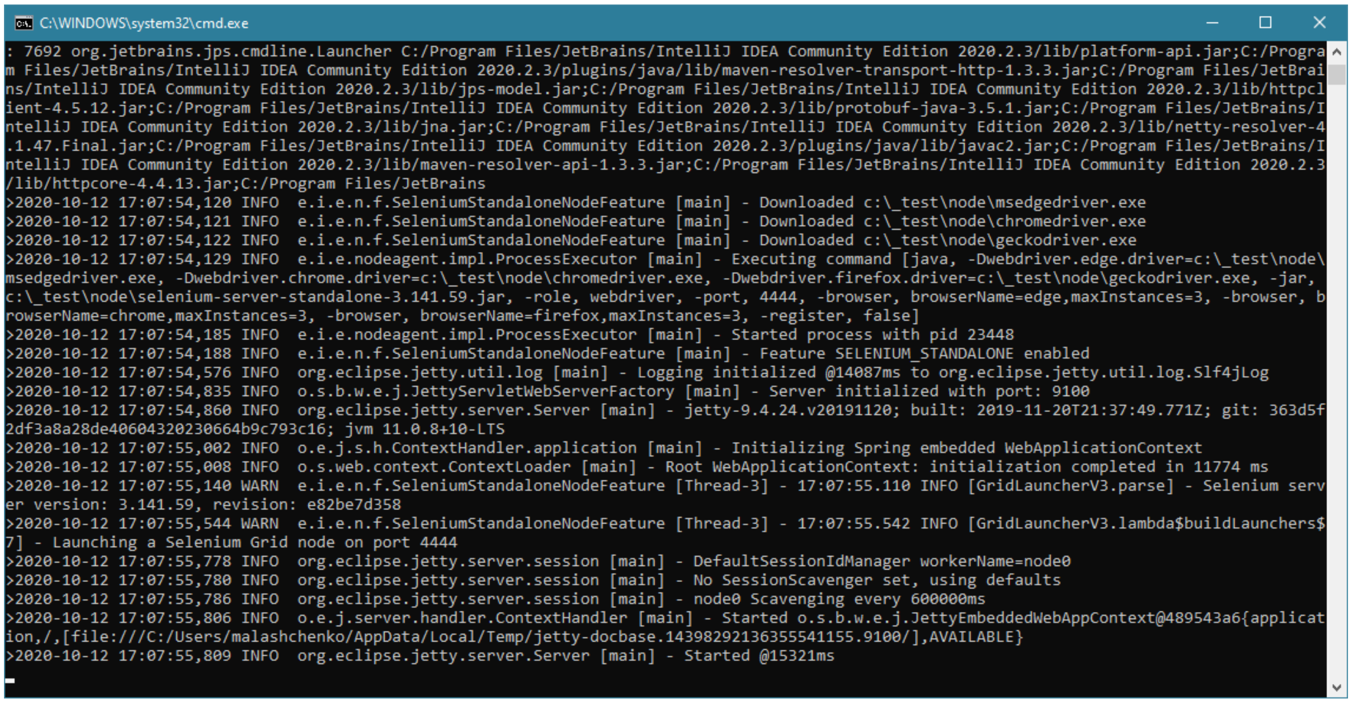
Go to the node's features tab and switch SELENIUM_STANDALONE on, with port 4444, and make it dedicated to protect you machine from CS runs:



Type	Status
APPIUM_NODE	Disabled
AP_RUN	Enabled
SELENIUM_HUB	Disabled
SELENIUM_NODE	Disabled
SELENIUM_STANDALONE	Enabled

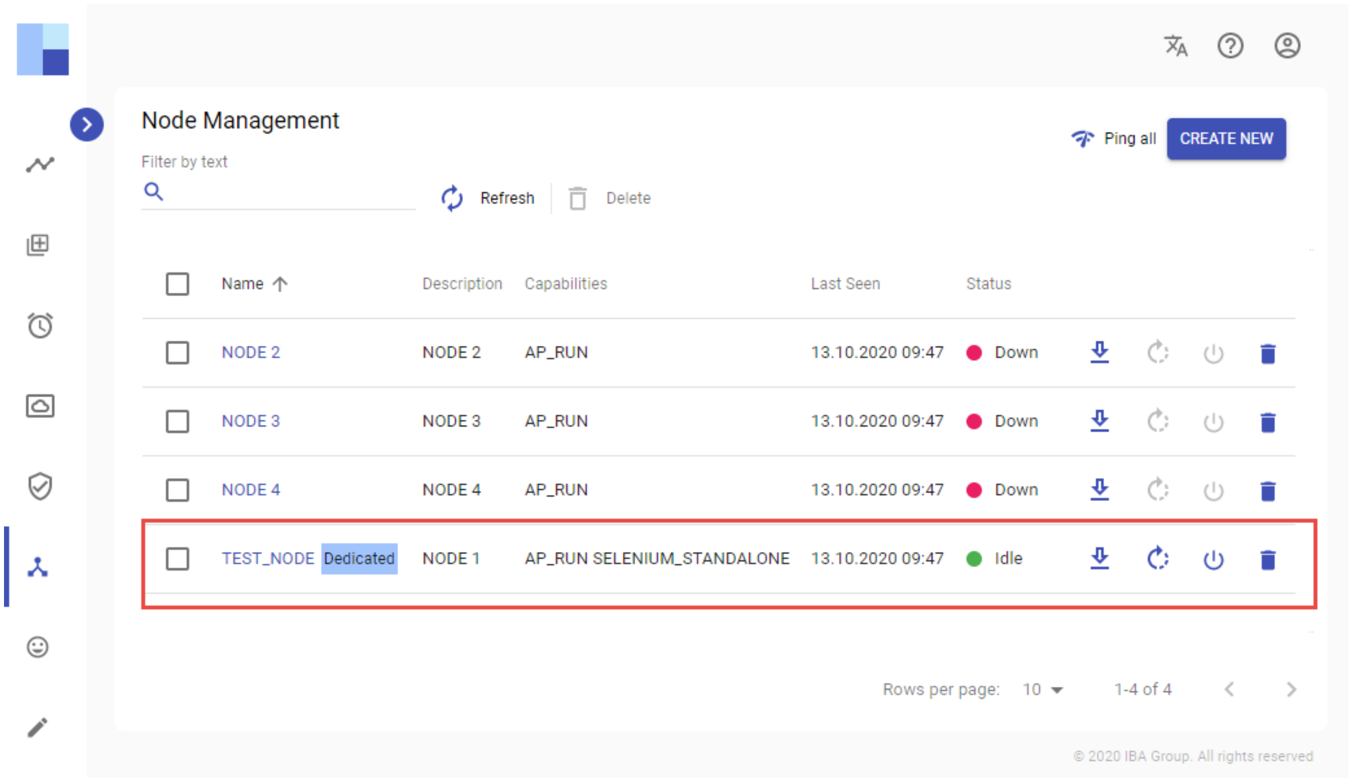
```
    "file": "selenium-server-standalone-3.141.59.jar",
    "drivers": {
      "edge": {
        "url": {
          "type": "S3",
          "location": "easyrpa/selenium/drivers/msedgedriver.exe"
        },
        "file": "msedgedriver.exe",
        "maxInstances": 3,
        "file_type": null
      },
      "chrome": {
        "url": {
          "type": "S3",
          "location": "easyrpa/selenium/drivers/chromedriver.exe"
        },
        "file": "chromedriver.exe",
        "maxInstances": 3,
        "file_type": null
      },
      "firefox": {
        "url": {
          "type": "S3",
          "location": "easyrpa/selenium/drivers/geckodriver.exe"
        },
        "file": "geckodriver.exe",
        "maxInstances": 3,
        "file_type": null
      }
    },
    "port": 4444
  }
```

Download node agent package on your machine, unzip and run node-agent.bat:



```
: 7692 org.jetbrains.jps.cmdline.Launcher C:/Program Files/JetBrains/IntelliJ IDEA Community Edition 2020.2.3/lib/platform-api.jar;C:/Progra
m Files/JetBrains/IntelliJ IDEA Community Edition 2020.2.3/plugins/java/lib/maven-resolver-transport-http-1.3.3.jar;C:/Program Files/JetBrai
ns/IntelliJ IDEA Community Edition 2020.2.3/lib/jps-model.jar;C:/Program Files/JetBrains/IntelliJ IDEA Community Edition 2020.2.3/lib/httpcl
ient-4.5.12.jar;C:/Program Files/JetBrains/IntelliJ IDEA Community Edition 2020.2.3/lib/protobuf-java-3.5.1.jar;C:/Program Files/JetBrains/I
ntelliJ IDEA Community Edition 2020.2.3/lib/jna.jar;C:/Program Files/JetBrains/IntelliJ IDEA Community Edition 2020.2.3/lib/netty-resolver-4
.1.47.Final.jar;C:/Program Files/JetBrains/IntelliJ IDEA Community Edition 2020.2.3/plugins/java/lib/javac2.jar;C:/Program Files/JetBrains/I
ntelliJ IDEA Community Edition 2020.2.3/lib/maven-resolver-api-1.3.3.jar;C:/Program Files/JetBrains/IntelliJ IDEA Community Edition 2020.2.3
/lib/httpcore-4.4.13.jar;C:/Program Files/JetBrains
>2020-10-12 17:07:54,120 INFO  e.i.e.n.f.SeleniumStandaloneNodeFeature [main] - Downloaded c:\_test\node\msedgedriver.exe
>2020-10-12 17:07:54,121 INFO  e.i.e.n.f.SeleniumStandaloneNodeFeature [main] - Downloaded c:\_test\node\chromedriver.exe
>2020-10-12 17:07:54,122 INFO  e.i.e.n.f.SeleniumStandaloneNodeFeature [main] - Downloaded c:\_test\node\geckodriver.exe
>2020-10-12 17:07:54,129 INFO  e.i.e.nodeagent.impl.ProcessExecutor [main] - Executing command [java, -Dwebdriver.edge.driver=c:\_test\node\
msedgedriver.exe, -Dwebdriver.chrome.driver=c:\_test\node\chromedriver.exe, -Dwebdriver.firefox.driver=c:\_test\node\geckodriver.exe, -jar,
c:\_test\node\selenium-server-standalone-3.141.59.jar, -role, webdriver, -port, 4444, -browser, browserName=edge,maxInstances=3, -browser, b
rowserName=chrome,maxInstances=3, -browser, browserName=firefox,maxInstances=3, -register, false]
>2020-10-12 17:07:54,185 INFO  e.i.e.nodeagent.impl.ProcessExecutor [main] - Started process with pid 23448
>2020-10-12 17:07:54,188 INFO  e.i.e.n.f.SeleniumStandaloneNodeFeature [main] - Feature SELENIUM_STANDALONE enabled
>2020-10-12 17:07:54,576 INFO  org.eclipse.jetty.util.log [main] - Logging initialized @14087ms to org.eclipse.jetty.util.log.Slf4jLog
>2020-10-12 17:07:54,835 INFO  o.s.b.w.e.j.JettyServletWebServerFactory [main] - Server initialized with port: 9100
>2020-10-12 17:07:54,860 INFO  org.eclipse.jetty.server.Server [main] - jetty-9.4.24.v20191120; built: 2019-11-20T21:37:49.771Z; git: 363d5f
2df3a8a28de40604320230664b9c793c16; jvm 11.0.8+10-LTS
>2020-10-12 17:07:55,002 INFO  o.e.j.s.h.ContextHandler.application [main] - Initializing Spring embedded WebApplicationContext
>2020-10-12 17:07:55,008 INFO  o.s.web.context.ContextLoader [main] - Root WebApplicationContext: initialization completed in 11774 ms
>2020-10-12 17:07:55,140 WARN  e.i.e.n.f.SeleniumStandaloneNodeFeature [Thread-3] - 17:07:55.110 INFO [GridLauncherV3.parse] - Selenium serv
er version: 3.141.59, revision: e82be7d358
>2020-10-12 17:07:55,544 WARN  e.i.e.n.f.SeleniumStandaloneNodeFeature [Thread-3] - 17:07:55.542 INFO [GridLauncherV3.lambda$buildLaunchers$
7] - Launching a Selenium Grid node on port 4444
>2020-10-12 17:07:55,778 INFO  org.eclipse.jetty.server.session [main] - DefaultSessionIdManager workerName=node0
>2020-10-12 17:07:55,780 INFO  org.eclipse.jetty.server.session [main] - No SessionScavenger set, using defaults
>2020-10-12 17:07:55,786 INFO  org.eclipse.jetty.server.session [main] - node0 Scavenging every 600000ms
>2020-10-12 17:07:55,806 INFO  o.e.j.server.handler.ContextHandler [main] - Started o.s.b.w.e.j.JettyEmbeddedWebAppContext@489543a6{applicat
ion,/,[file:///C:/Users/malashchenko/AppData/Local/Temp/jetty-docbase.14398292136355541155.9100/],AVAILABLE}
>2020-10-12 17:07:55,809 INFO  org.eclipse.jetty.server.Server [main] - Started @15321ms
```

The node becomes available for CS:



Node Management

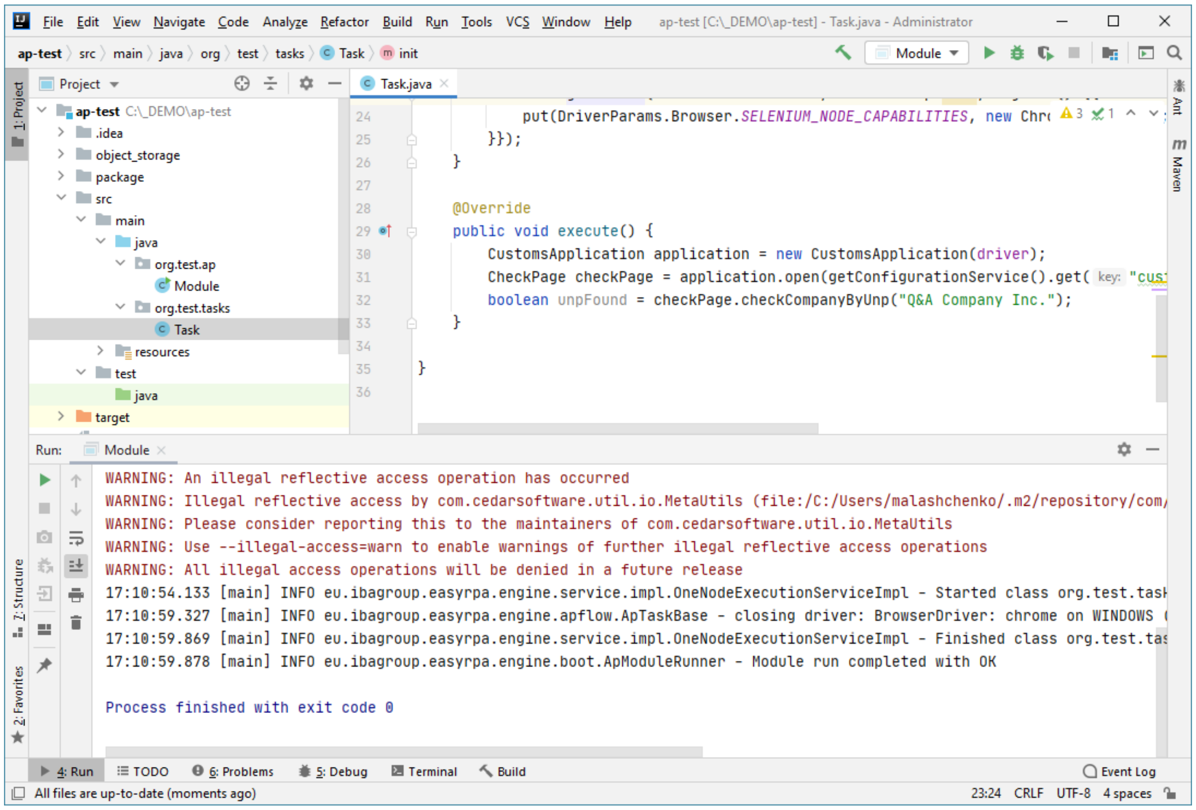
Filter by text

	Name	Description	Capabilities	Last Seen	Status
☐	NODE 2	NODE 2	AP_RUN	13.10.2020 09:47	● Down
☐	NODE 3	NODE 3	AP_RUN	13.10.2020 09:47	● Down
☐	NODE 4	NODE 4	AP_RUN	13.10.2020 09:47	● Down
☐	TEST_NODE Dedicated	NODE 1	AP_RUN SELENIUM_STANDALONE	13.10.2020 09:47	● Idle

Rows per page: 10 1-4 of 4

© 2020 IBA Group. All rights reserved

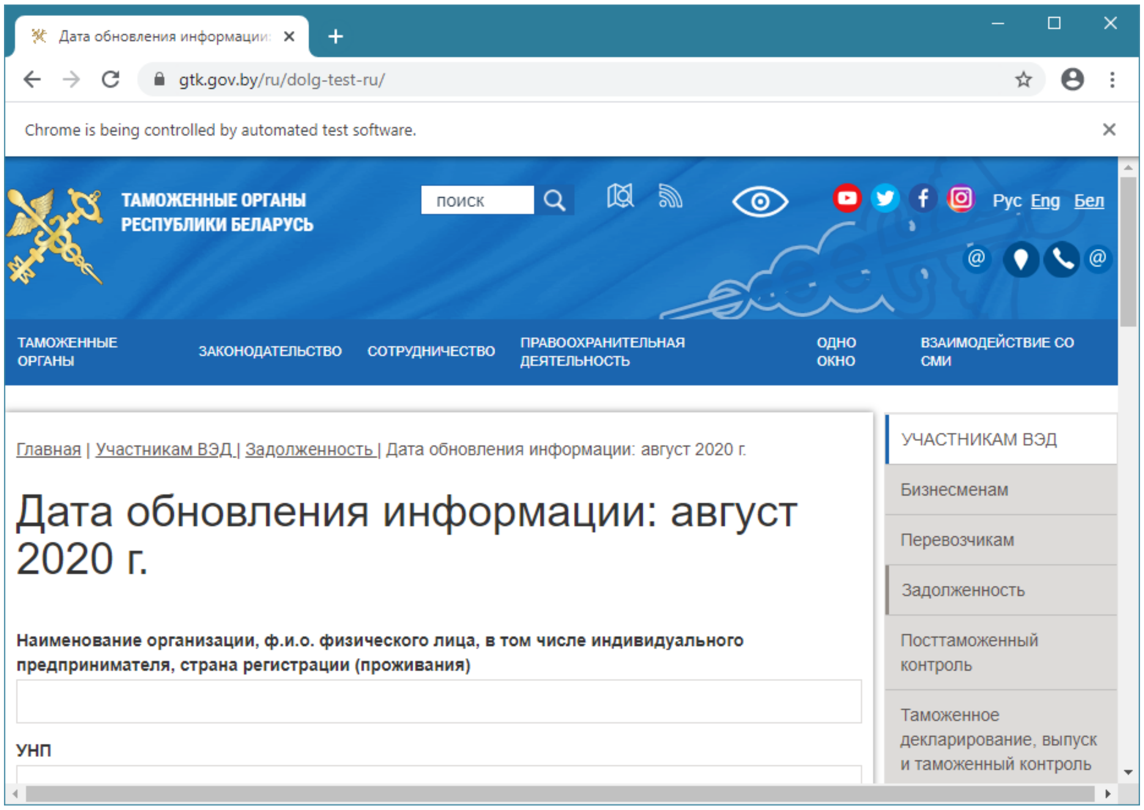
Now you can run the AP using local senenim connection of your easyRPA node:


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help ap-test [C:_DEMO\ap-test] - Task.java - Administrator
ap-test src main java org test tasks Task init
Module
Project
ap-test C:_DEMO\ap-test
.idea
object_storage
package
src
main
java
org.test.ap
Module
org.test.tasks
Task
resources
test
java
target
Task.java
Run: Module
4: Run
TODO
6: Problems
5: Debug
Terminal
Build
Event Log
All files are up-to-date (moments ago)
23:24 CRLF UTF-8 4 spaces

```
            put(DriverParams.Browser.SELENIUM_NODE_CAPABILITIES, new Chr
        }});
    }

    @Override
    public void execute() {
        CustomsApplication application = new CustomsApplication(driver);
        CheckPage checkPage = application.open(getConfigurationService().get( key: "cus
        boolean unpFound = checkPage.checkCompanyByUnp("Q&A Company Inc.");
    }

}
```

```
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by com.cedarsoftware.util.io.MetaUtils (file:/C:/Users/malashchenko/.m2/repository/com/
WARNING: Please consider reporting this to the maintainers of com.cedarsoftware.util.io.MetaUtils
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
17:10:54.133 [main] INFO eu.ibagroup.easyrpa.engine.service.impl.OneNodeExecutionServiceImpl - Started class org.test.task
17:10:59.327 [main] INFO eu.ibagroup.easyrpa.engine.apflow.ApTaskBase - closing driver: BrowserDriver: chrome on WINDOWS (
17:10:59.869 [main] INFO eu.ibagroup.easyrpa.engine.service.impl.OneNodeExecutionServiceImpl - Finished class org.test.tas
17:10:59.878 [main] INFO eu.ibagroup.easyrpa.engine.boot.ApModuleRunner - Module run completed with OK

Process finished with exit code 0
```

Robot does RPA tasks on your developer machine.


Дата обновления информации

gtk.gov.by/ru/dolg-test-ru/

Chrome is being controlled by automated test software.

ТАМОЖЕННЫЕ ОРГАНЫ РЕСПУБЛИКИ БЕЛАРУСЬ

поиск

Рус Eng Бел

ТАМОЖЕННЫЕ ОРГАНЫ | ЗАКОНОДАТЕЛЬСТВО | СОТРУДНИЧЕСТВО | ПРАВООХРАНИТЕЛЬНАЯ ДЕЯТЕЛЬНОСТЬ | ОДНО ОКНО | ВЗАИМОДЕЙСТВИЕ СО СМИ

Главная | Участникам ВЭД | Задолженность | Дата обновления информации: август 2020 г.

Дата обновления информации: август 2020 г.

Наименование организации, ф.и.о. физического лица, в том числе индивидуального предпринимателя, страна регистрации (проживания)

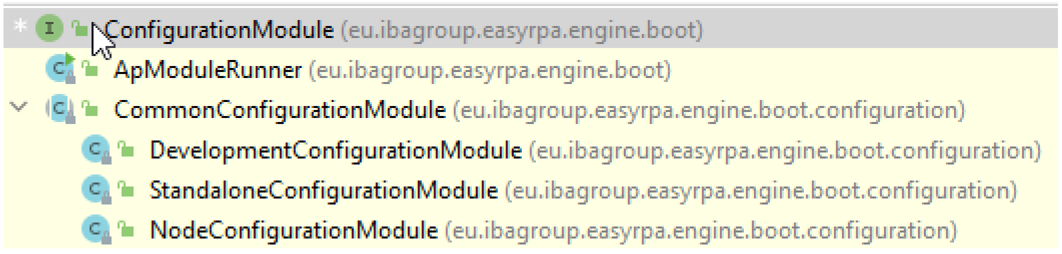
УНП

УЧАСТНИКАМ ВЭД
Бизнесменам
Перевозчикам
Задолженность
Посттаможенный контроль
Таможенное декларирование, выпуск и таможенный контроль

Setup module run configurations

There are 2 possible Automation Process run configurations available for development:

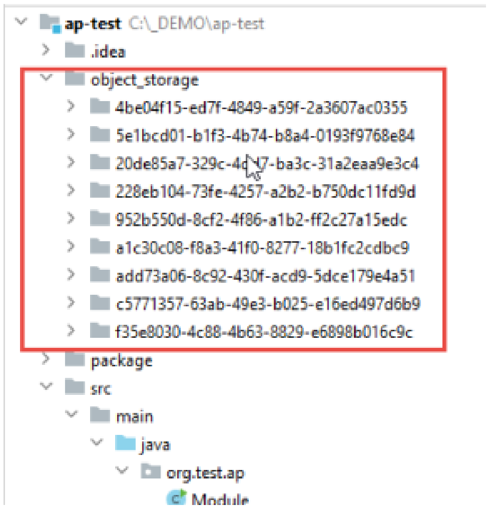
- StandaloneConfigurationModule
- DevelopmentConfigurationModule


ConfigurationModule (eu.ibagroup.easyrpa.engine.boot)
ApModuleRunner (eu.ibagroup.easyrpa.engine.boot)
CommonConfigurationModule (eu.ibagroup.easyrpa.engine.boot.configuration)
DevelopmentConfigurationModule (eu.ibagroup.easyrpa.engine.boot.configuration)
StandaloneConfigurationModule (eu.ibagroup.easyrpa.engine.boot.configuration)
NodeConfigurationModule (eu.ibagroup.easyrpa.engine.boot.configuration)

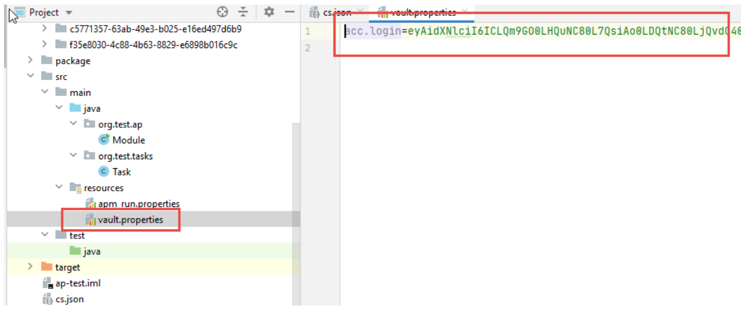
StandaloneConfigurationModule

This is default configuration. Its main features:

- task history is stored on file system


```
ap-test C:\_DEMO\ap-test
  .idea
  object_storage
    4be04f15-ed7f-4849-a59f-2a3607ac0355
    5e1bcd01-b1f3-4b74-b8a4-0193f9768e84
    20de85a7-329c-4d 7-ba3c-31a2eaa9e3c4
    228eb104-73fe-4257-a2b2-b750dc11fd9d
    952b550d-8cf2-4f86-a1b2-ff2c27a15edc
    a1c30c08-f8a3-41f0-8277-18b1fc2cdbc9
    add73a06-8c92-430f-acd9-5dce179e4a51
    c5771357-63ab-49e3-b025-e16ed497d6b9
    f35e8030-4c88-4b63-8829-e6898b016c9c
  package
  src
    main
      java
        org.test.ap
          Module
```

- local logs only
- local vault storage from property file



```
c5771357-63ab-49e3-b025-e16ed497d6b9
f35e8030-4c88-4b63-8829-e6898b016c9c
package
src
  main
    java
      org.test.ap
        Module
      org.test.tasks
        Task
    resources
      apm_run.properties
      vault.properties
  test
    java
target
ap-test.iml
cs.json
```

```
acc.login=eyAidXNlciI6ICLQm9GO0LHQuNC80L7QsiAo0LDQtNC80LjQvd040
```

- *datastore is emulated in memory and does not support SQL query*

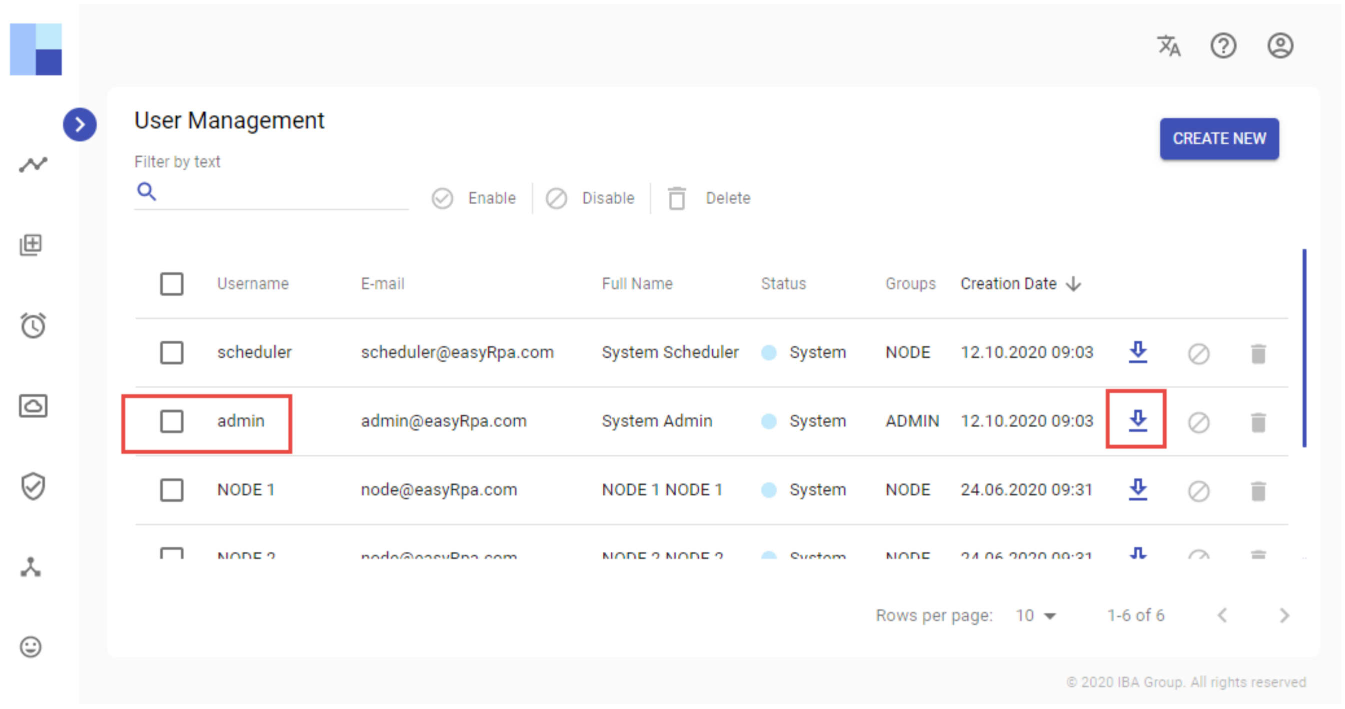
DevelopmentConfigurationModule

This configuration is intended to make AP runs in the same environment that ControlServer provides. Its main features are:

- task history are stored on file system
- local logs only
- local vault storage from property file
- **datastore are hosted on Control Server**

To use this configuration you should link your run to a ControlServer instance.

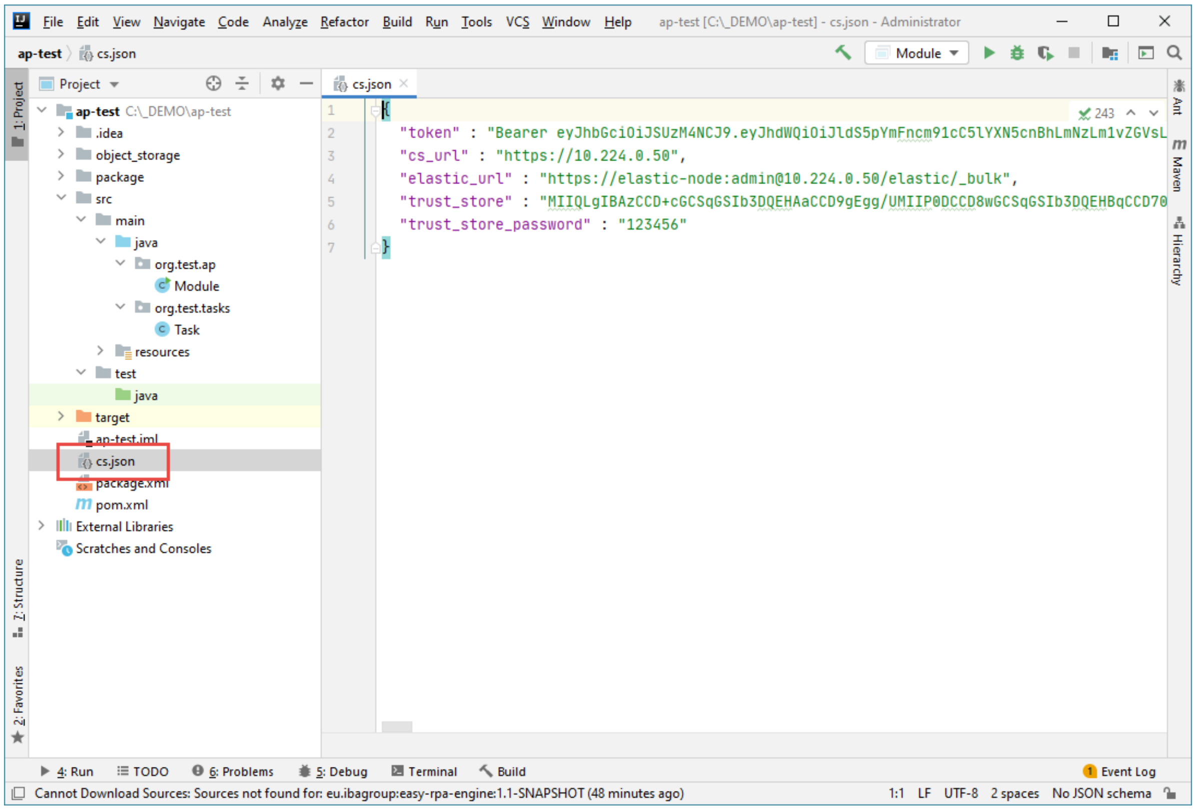
Go to ControlServer users page and download development configuration JSON file, put it into run working directory.



User Management

	Username	E-mail	Full Name	Status	Groups	Creation Date
☐	scheduler	scheduler@easyRpa.com	System Scheduler	System	NODE	12.10.2020 09:03
☐	admin	admin@easyRpa.com	System Admin	System	ADMIN	12.10.2020 09:03
☐	NODE 1	node@easyRpa.com	NODE 1 NODE 1	System	NODE	24.06.2020 09:31
☐	NODE 2	node@easyRpa.com	NODE 2 NODE 2	System	NODE	24.06.2020 09:31

Rename file to **cs.json** and put it into module working directory:

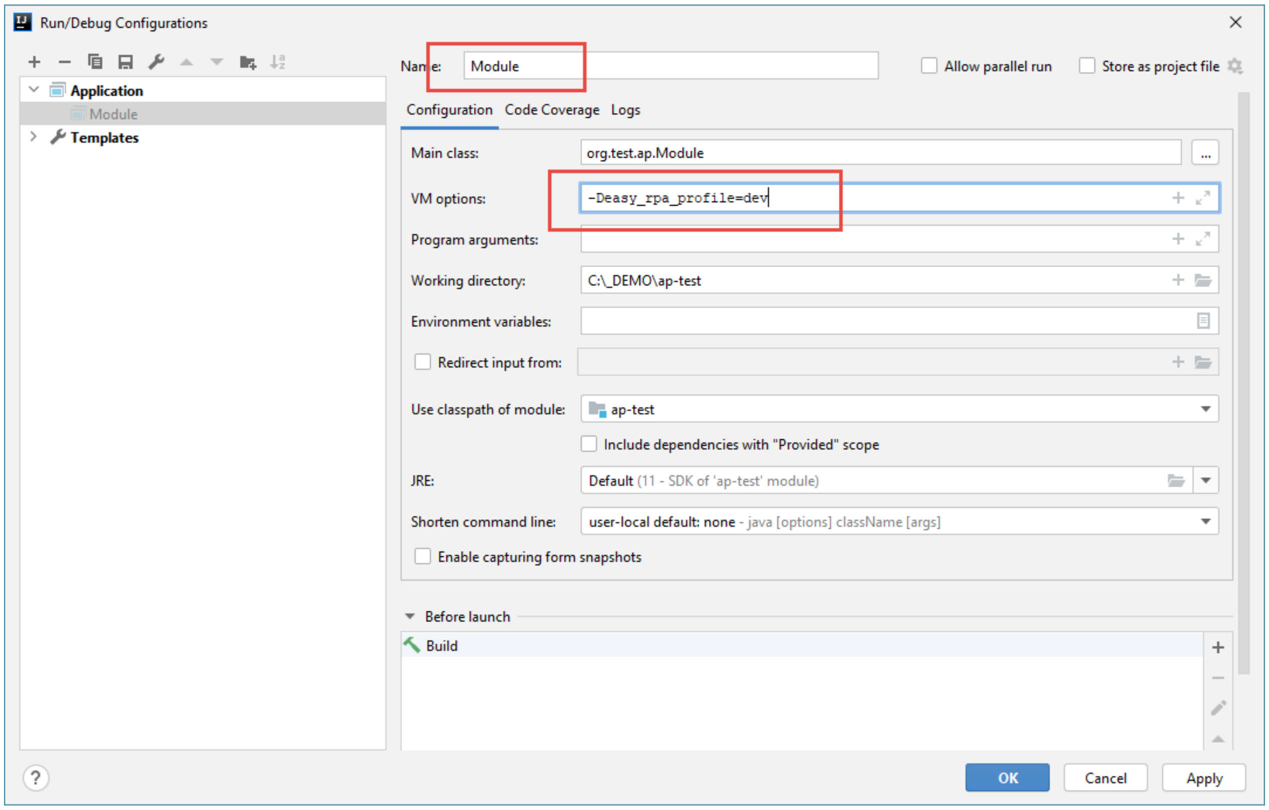


```
{
  "token" : "Bearer eyJhbGciOiJSUzM4NCJ9.eyJhdWQiOiJldS5pYmFncm91cC5lYXN5cnBhLmNzLm1vZGVsL
  "cs_url" : "https://10.224.0.50",
  "elastic_url" : "https://elastic-node:admin@10.224.0.50/elastic/_bulk",
  "trust_store" : "MIIQLgIBAzCCD+cGCSqGSIb3DQEHAaCCD9gEgg/UMIIP0DCCD8wGCSqGSIb3DQEHBqCCD70
  "trust_store_password" : "123456"
}
```

Cannot Download Sources: Sources not found for: eu.ibagroup:easy-rpa-engine:1.1-SNAPSHOT (48 minutes ago)

Add java system property that defines easyRPA profile to use in your launch configuration:

```
-Deasy_rpa_profile=dev
```


Run/Debug Configurations

Application

Module

Templates

Name: Module

Allow parallel run

Store as project file

Configuration Code Coverage Logs

Main class: org.test.ap.Module

VM options: -Deasy_rpa_profile=dev

Program arguments:

Working directory: C:_DEMO\ap-test

Environment variables:

Redirect input from:

Use classpath of module: ap-test

Include dependencies with "Provided" scope

JRE: Default (11 - SDK of 'ap-test' module)

Shorten command line: user-local default: none - java [options] className [args]

Enable capturing form snapshots

Before launch

Build

OK Cancel Apply

Run Automation Process in cooperate with Control Server

Use **DevelopmentConfigurationModule** you created in previous section and add the following classes:

Book

```
package org.test.domain;

import eu.ibagroup.easyrpa.persistence.annotations.Column;
import eu.ibagroup.easyrpa.persistence.annotations.CreatePolicy;
import eu.ibagroup.easyrpa.persistence.annotations.Entity;
import eu.ibagroup.easyrpa.persistence.annotations.Id;
import lombok.Data;
import lombok.NoArgsConstructor;

@Entity(value = "crud_example_books", create = CreatePolicy.IF_ABSENT)
@NoArgsConstructor
@Data // will generate getters, setters and toString
public class Book {

    @Id
    @Column("index")
    private String index;

    @Column("name")
    private String name;

    @Column("author")
    private String author;

    public Book(String index, String name, String author) {
        this.index = index;
        this.name = name;
        this.author = author;
    }
}
```

BookRepository

```
package org.test.ap.repository;

import eu.ibagroup.easyrpa.persistence.CrudRepository;
import org.test.domain.Book;

import java.util.ArrayList;
import java.util.List;

public interface BookRepository extends CrudRepository<Book, String> {

    //Find books by name
    default List<Book> getBookByName(String name) {

        List<Book> result = new ArrayList<>();
        for (Book book : findAll()) {
            if (book.getName().equals(name)) {
                result.add(book);
            }
        }
        return result;
    }

    //Save list of books
    default void saveBooks(List<Book> books) {

        for (Book book : books) {
            save(book);
        }
    }
}
```

SampleTask

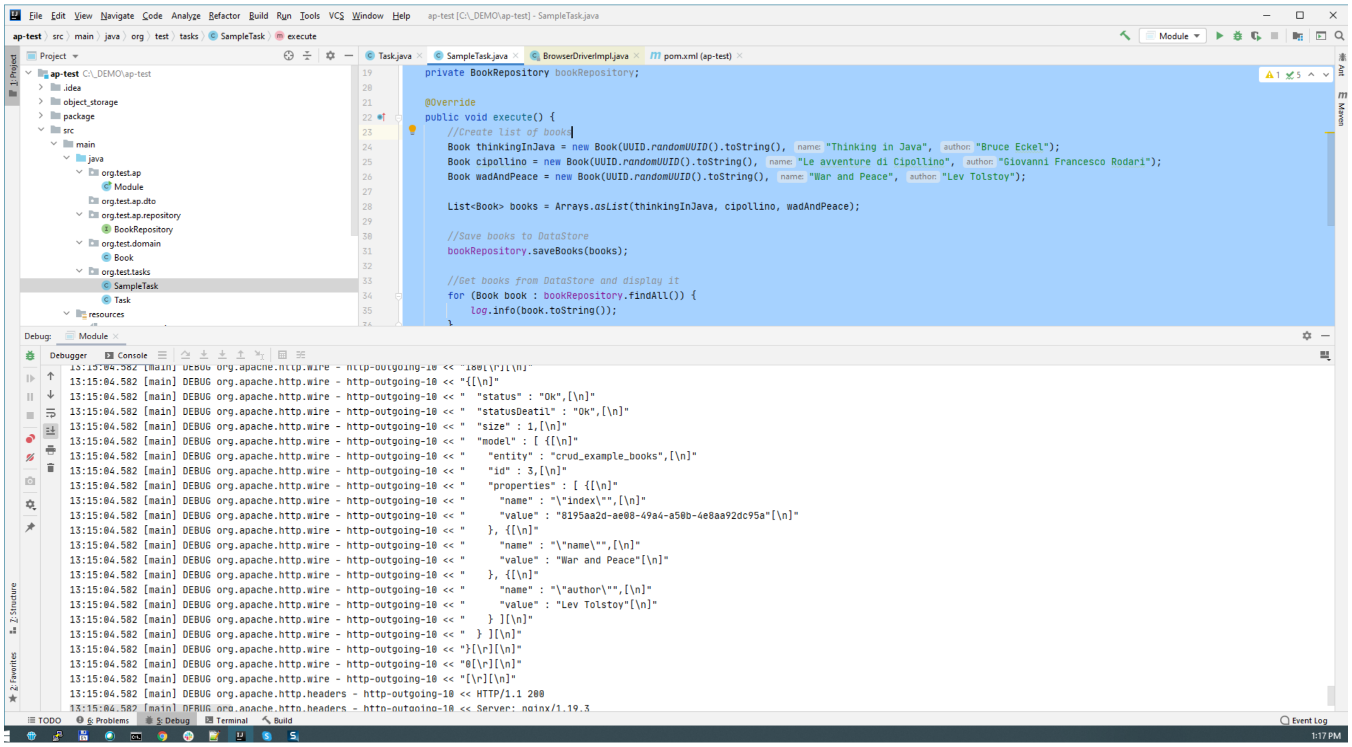
```
package org.test.tasks;

import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApTaskBase;
import lombok.extern.slf4j.Slf4j;
import org.test.ap.repository.BookRepository;
import org.test.domain.Book;

package org.test.tasks;

import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApTaskBase;
import lombok.extern.slf4j.Slf4j;
import org.test.ap.repository.BookRepository;
import org.test.domain.Book;

import javax.inject.Inject;
import java.util.Arrays;
import java.util.List;
import java.util.UUID;

@ApTaskEntry(name = "Sample Task")
@Slf4j
public class SampleTask extends ApTaskBase {

    @Inject
    private BookRepository bookRepository;

    @Override
    public void execute() throws Exception {
        //Create list of books
        Book thinkingInJava = new Book(UUID.randomUUID().toString(), "Thinking in Java", "Bruce Eckel");
        Book cipollino = new Book(UUID.randomUUID().toString(), "Le avventure di Cipollino", "Giovanni
Francesco Rodari");
        Book wadAndPeace = new Book(UUID.randomUUID().toString(), "War and Peace", "Lev Tolstoy");

        List<Book> books = Arrays.asList(thinkingInJava, cipollino, wadAndPeace);

        //Save books to DataStore
        bookRepository.saveBooks(books);

        //Get books from DataStore and display it
        for (Book book : bookRepository.findAll()) {
            log.info(book.toString());
        }

        //Find book by name
        for (Book book : bookRepository.getBookByName("Thinking in Java")) {
            log.info(book.toString());
        }

        Thread.sleep(120_000L);
        //Clean repository
        for (Book book : bookRepository.findAll()) {
            bookRepository.delete(book);
        }

    }
}
```

Change your module to call SampleTask:

Module

```java
package org.test.ap;

import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModule;
import eu.ibagroup.easyrpa.engine.apflow.ApModuleBase;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import eu.ibagroup.easyrpa.engine.boot.ApModuleRunner;
import lombok.extern.slf4j.Slf4j;
import org.test.tasks.SampleTask;

@Slf4j
@ApModuleEntry(name = "Sample Process", description = "Sample datastore description")
public class Module extends ApModuleBase implements ApModule {

    public TaskOutput run() throws Exception {
        return execute(getInput(), SampleTask.class);
    }

    public static void main(String[] arg) {
        ApModuleRunner runner = new ApModuleRunner();
        runner.launch(Module.class);
    }
}
```

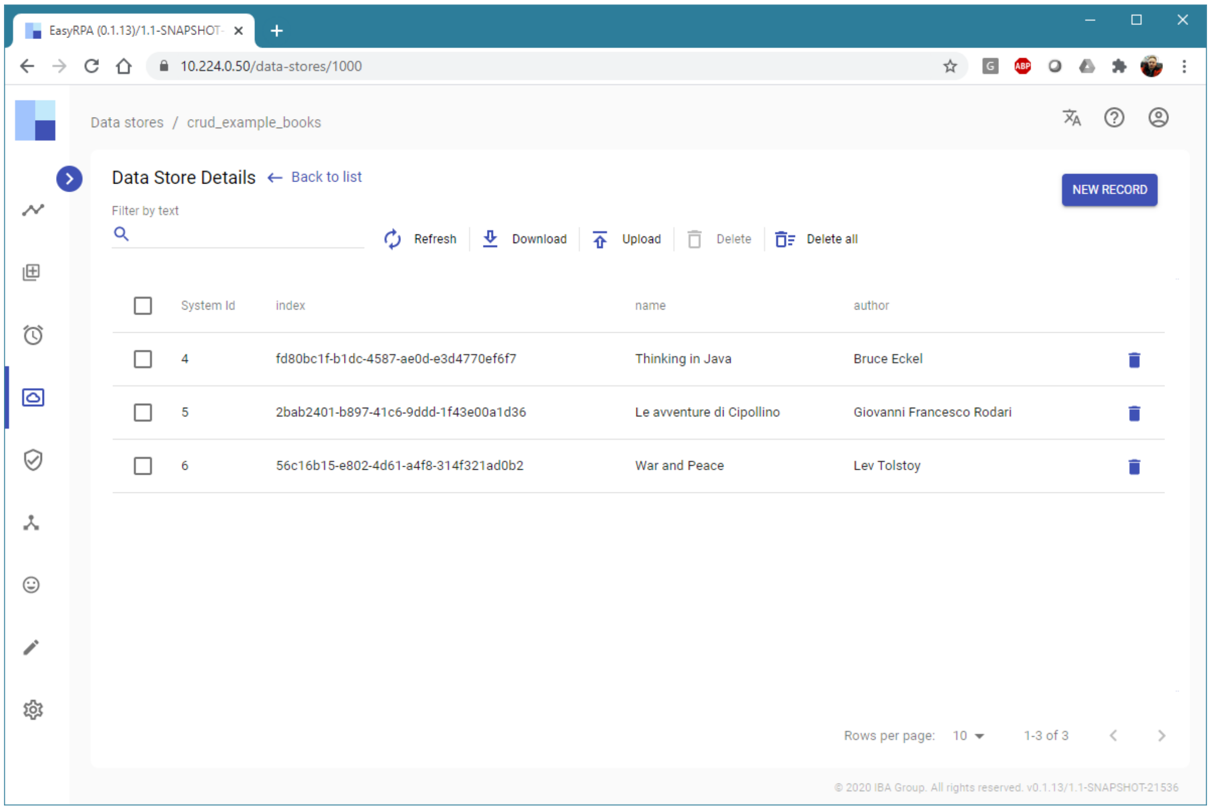
Run the Module:



```java
    private BookRepository bookRepository;

    @Override
    public void execute() {
        //Create list of books
        Book thinkingInJava = new Book(UUID.randomUUID().toString(), name: "Thinking in Java", author: "Bruce Eckel");
        Book cipollino = new Book(UUID.randomUUID().toString(), name: "Le avventure di Cipollino", author: "Giovanni Francesco Rodari");
        Book wadAndPeace = new Book(UUID.randomUUID().toString(), name: "War and Peace", author: "Lev Tolstoy");

        List<Book> books = Arrays.asList(thinkingInJava, cipollino, wadAndPeace);

        //Save books to DataStore
        bookRepository.saveBooks(books);

        //Get books from DataStore and display it
        for (Book book : bookRepository.findAll()) {
            log.info(book.toString());
```

```
13:15:04.582 [main] DEBUG org.apache.http.wire - http-outgoing-10 << "180[\r][\n]"
13:15:04.582 [main] DEBUG org.apache.http.wire - http-outgoing-10 << "{[\n]"
13:15:04.582 [main] DEBUG org.apache.http.wire - http-outgoing-10 << "  "status" : "Ok",[\n]"
13:15:04.582 [main] DEBUG org.apache.http.wire - http-outgoing-10 << "  "statusDeatil" : "Ok",[\n]"
13:15:04.582 [main] DEBUG org.apache.http.wire - http-outgoing-10 << "  "size" : 1,[\n]"
13:15:04.582 [main] DEBUG org.apache.http.wire - http-outgoing-10 << "  "model" : [ {[\n]"
13:15:04.582 [main] DEBUG org.apache.http.wire - http-outgoing-10 << "    "entity" : "crud_example_books",[\n]"
13:15:04.582 [main] DEBUG org.apache.http.wire - http-outgoing-10 << "    "id" : 3,[\n]"
13:15:04.582 [main] DEBUG org.apache.http.wire - http-outgoing-10 << "    "properties" : [ {[\n]"
13:15:04.582 [main] DEBUG org.apache.http.wire - http-outgoing-10 << "      "name" : "\"index\"",[\n]"
13:15:04.582 [main] DEBUG org.apache.http.wire - http-outgoing-10 << "      "value" : "8195aa2d-ae08-49a4-a50b-4e8aa92dc95a"[\n]"
13:15:04.582 [main] DEBUG org.apache.http.wire - http-outgoing-10 << "    }, {[\n]"
13:15:04.582 [main] DEBUG org.apache.http.wire - http-outgoing-10 << "      "name" : "\"name\"",[\n]"
13:15:04.582 [main] DEBUG org.apache.http.wire - http-outgoing-10 << "      "value" : "War and Peace"[\n]"
13:15:04.582 [main] DEBUG org.apache.http.wire - http-outgoing-10 << "    }, {[\n]"
13:15:04.582 [main] DEBUG org.apache.http.wire - http-outgoing-10 << "      "name" : "\"author\"",[\n]"
13:15:04.582 [main] DEBUG org.apache.http.wire - http-outgoing-10 << "      "value" : "Lev Tolstoy"[\n]"
13:15:04.582 [main] DEBUG org.apache.http.wire - http-outgoing-10 << "    } ][\n]"
13:15:04.582 [main] DEBUG org.apache.http.wire - http-outgoing-10 << "  } ][\n]"
13:15:04.582 [main] DEBUG org.apache.http.wire - http-outgoing-10 << "}[\r][\n]"
13:15:04.582 [main] DEBUG org.apache.http.wire - http-outgoing-10 << "0[\r][\n]"
13:15:04.582 [main] DEBUG org.apache.http.wire - http-outgoing-10 << "[\r][\n]"
13:15:04.582 [main] DEBUG org.apache.http.headers - http-outgoing-10 << HTTP/1.1 200
13:15:04.582 [main] DEBUG org.apache.http.headers - http-outgoing-10 << Server: nginx/1.19.3
```

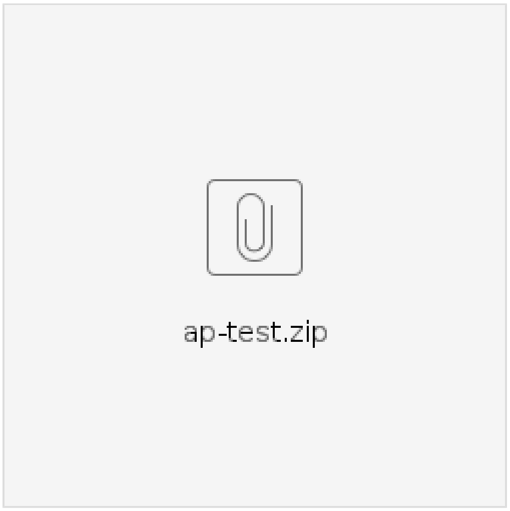
Check results in control server DataStore:


EasyRPA (0.1.13)/1.1-SNAPSHOT-
10.224.0.50/data-stores/1000
Data stores / crud_example_books
Data Store Details
Back to list
NEW RECORD
Filter by text
Refresh
Download
Upload
Delete
Delete all

	System Id	index	name	author
☐	4	fd80bc1f-b1dc-4587-ae0d-e3d4770ef6f7	Thinking in Java	Bruce Eckel
☐	5	2bab2401-b897-41c6-9ddd-1f43e00a1d36	Le avventure di Cipollino	Giovanni Francesco Rodari
☐	6	56c16b15-e802-4d61-a4f8-314f321ad0b2	War and Peace	Lev Tolstoy

Rows per page: 10
1-3 of 3
© 2020 IBA Group. All rights reserved. v0.1.13/1.1-SNAPSHOT-21536

Here is the project archive for testing:



Deploy on control server

There are two ways to deploy your Automation Process on Control Server:

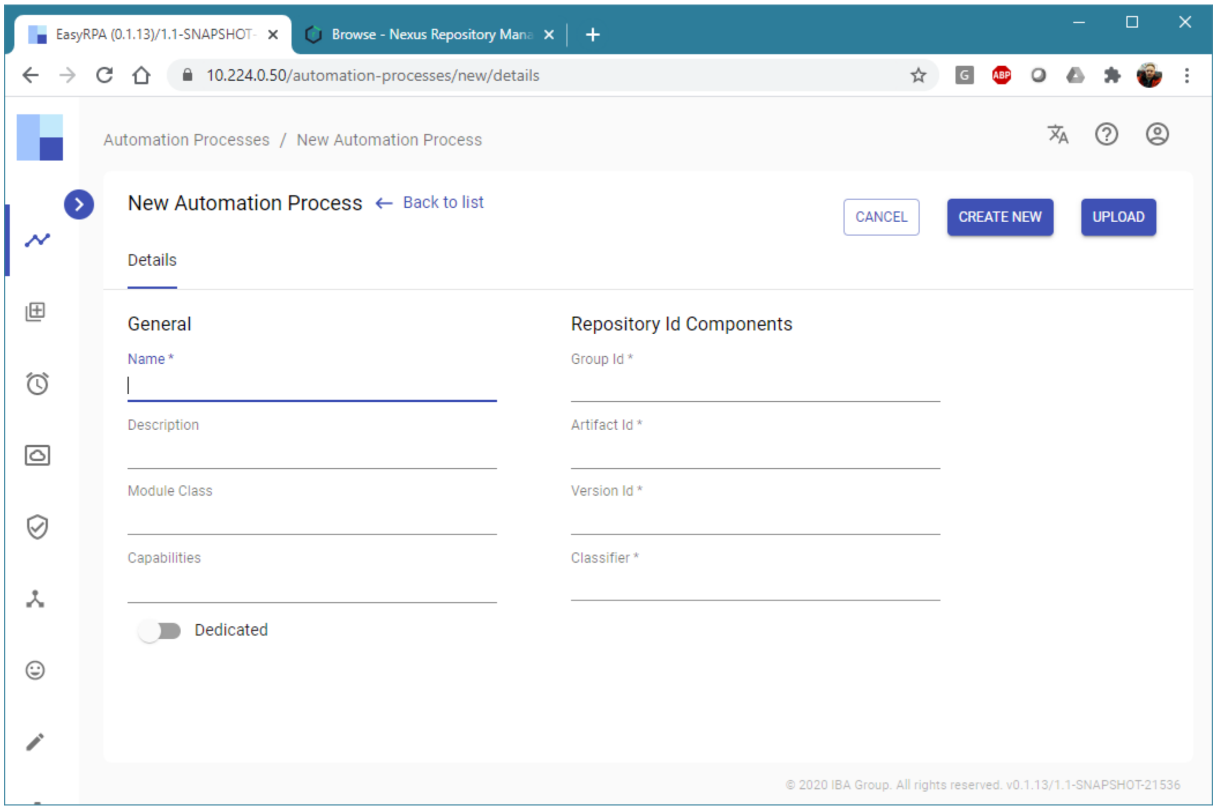
1. Deploy on Control Server manually
2. Upload AP zip package

Deploy on Control Server manually

Deploy AP jar on easyRPA nexus. for this you can run maven deploy with **nexus-deploy** profile:

```
mvn clean deploy -P nexus-deploy
```

Then go to control server and define your AP, Secrets, Datastores:


EasyRPA (0.1.13)/1.1-SNAPSHOT-
Browse - Nexus Repository Mana
10.224.0.50/automation-processes/new/details
Automation Processes / New Automation Process
New Automation Process ← Back to list
CANCEL
CREATE NEW
UPLOAD
Details

General

Name *

Description

Module Class

Capabilities

Dedicated

Repository Id Components

Group Id *

Artifact Id *

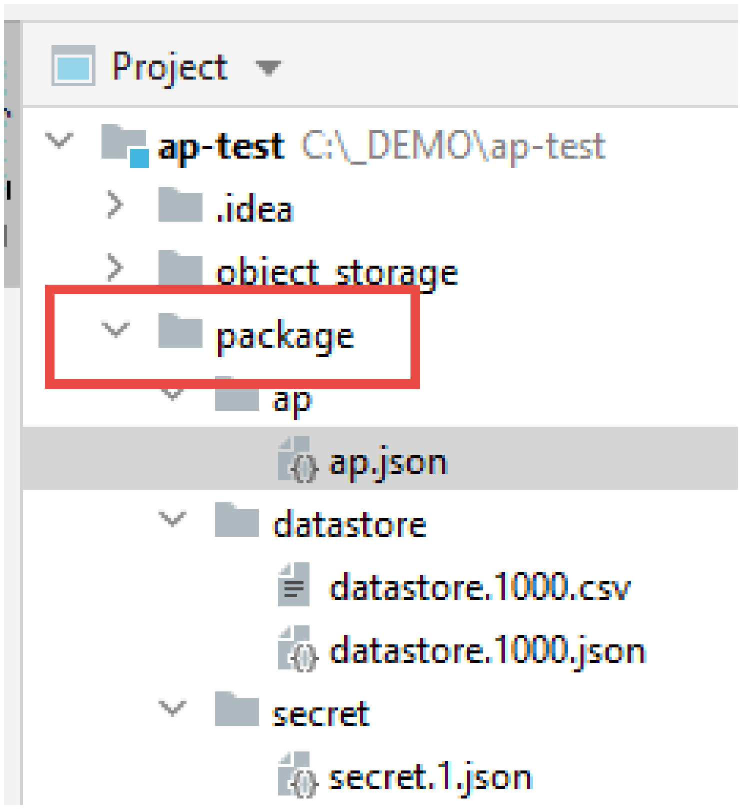
Version Id *

Classifier *

© 2020 IBA Group. All rights reserved. v0.1.13/1.1-SNAPSHOT-21536

Upload AP zip package

There is a possibility to import AutomationProcesses, Datastore, Secrets using package zip file. You need to define it structure in the package folder:



```
Project
ap-test C:\_DEMO\ap-test
  .idea
  object_storage
  package
    ap
      ap.json
    datastore
      datastore.1000.csv
      datastore.1000.json
    secret
      secret.1.json
```

Refer to the package definition article for details.

The **local-package** profile provides AP zip for CS uploading, it can be generated with command:

```
mvn clean deploy -P local-package
```

AP Export maven plugin

The cs-upload profile can be used for CI/CD builds. It contains **maven-assembly-plugin** that uploads the generated package to CS.

(v. 2.2) Automation Process Project structure

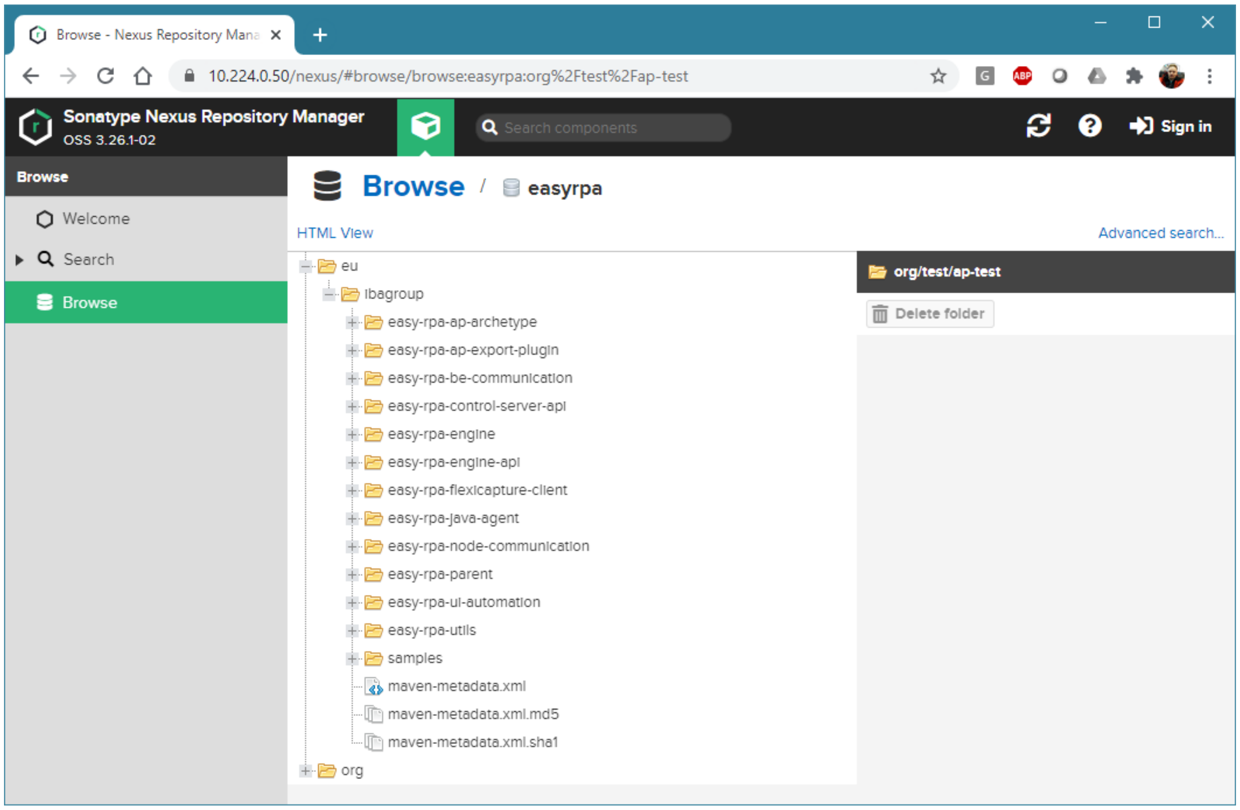
Content

Dependencies

All stuff you need to use during development process are coming together with Control Server installation in onboard nexus:

Nexus URL

https://<cs url>/nexus/#browse/browse:easyrpa:org%2Ftest%2Fap-test



Project generation

Project can be generated from archetype using either CLI or IDE.

CLI

The following command creates a project from archetype:

```
mvn archetype:generate -DarchetypeGroupId=eu.ibagroup -DarchetypeArtifactId=easy-rpa-ap-archetype -
DarchetypeVersion=1.1-SNAPSHOT -DgroupId=eu.ibagroup -DartifactId=test -Dversion=0.0.1-SNAPSHOT
```

As a result the stub project for eu.ibagroup:test:0.0.1-SNAPSHOT will be generated

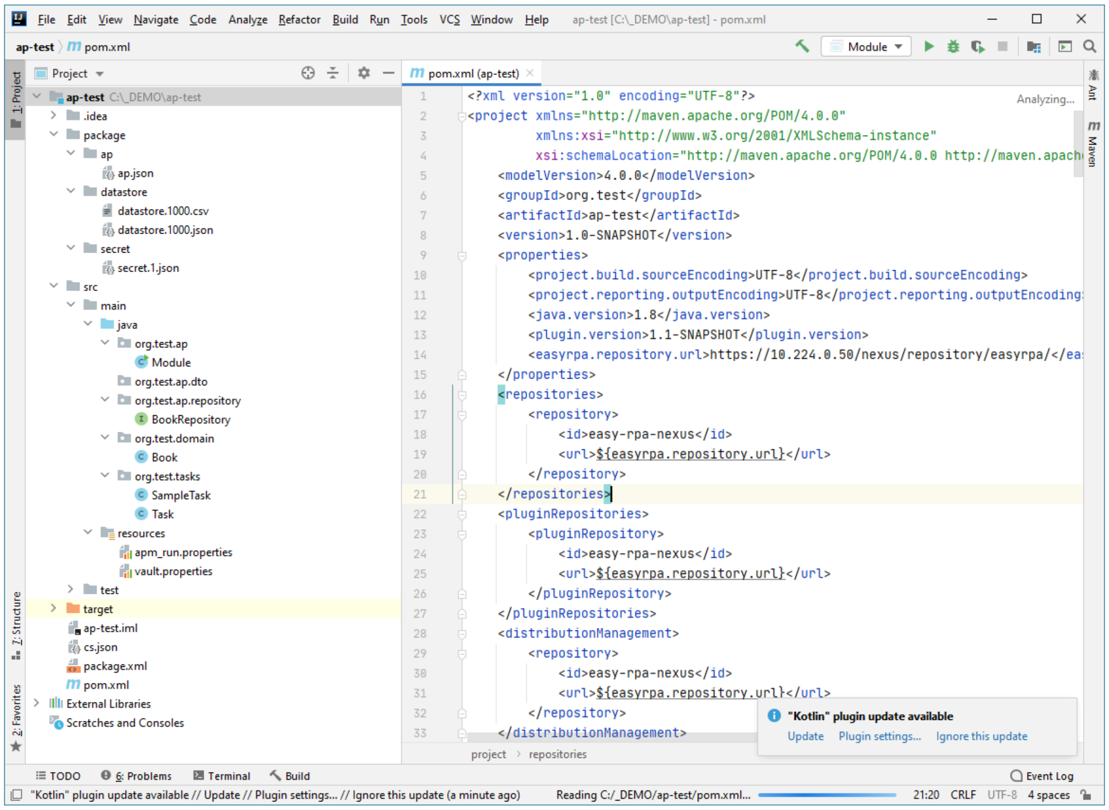
IDE

First, add remote catalog with archetype *easy-rpa-ap-archetype* in your IDE

Second, use the sequence "New project Maven project", then choose *easy-rpa-ap-archetype* as the base archetype to generate project from.

Project structure

A sample project for abstract Automation Process has the following file structure:



```
ap-test C:\_DEMO\ap-test
  .idea
  package
    ap
      ap.json
    datastore
      datastore.1000.csv
      datastore.1000.json
    secret
      secret.1.json
  src
    main
      java
        org.test.ap
          Module
        org.test.ap.dto
        org.test.ap.repository
          BookRepository
        org.test.domain
          Book
        org.test.tasks
          SampleTask
          Task
      resources
        apm_run.properties
        vault.properties
    test
  target
  ap-test.iml
  cs.json
  package.xml
  pom.xml
External Libraries
Scratches and Consoles
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apach
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.test</groupId>
    <artifactId>ap-test</artifactId>
    <version>1.0-SNAPSHOT</version>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding
        <java.version>1.8</java.version>
        <plugin.version>1.1-SNAPSHOT</plugin.version>
        <easyrpa.repository.url>https://10.224.0.50/nexus/repository/easyrpa/</ea
    </properties>
    <repositories>
        <repository>
            <id>easy-rpa-nexus</id>
            <url>${easyrpa.repository.url}</url>
        </repository>
    </repositories>
    <pluginRepositories>
        <pluginRepository>
            <id>easy-rpa-nexus</id>
            <url>${easyrpa.repository.url}</url>
        </pluginRepository>
    </pluginRepositories>
    <distributionManagement>
        <repository>
            <id>easy-rpa-nexus</id>
            <url>${easyrpa.repository.url}</url>
        </repository>
    </distributionManagement>
```

"Kotlin" plugin update available
Update   Plugin settings...   Ignore this update

"Kotlin" plugin update available // Update // Plugin settings... // Ignore this update (a minute ago)   Reading C:/_DEMO/ap-test/pom.xml...   21:20   CRLF   UTF-8   4 spaces

Folder	Content	File extensions
package/ap	contains definitions of Automation Processes	.json

package /datastore	contains datastore-related sources: 1) datastore definition (mandatory) 2) initial data in the form of csv file (optional) 3) metadata describing format of csv file (optional) All files related to the same datastore must have the same names and can differ only in file extension. For example, the following files describe different features of one datastore : datastore/ds.1000.json datastore/ds.1000.csv	.json .csv .format
package/secret	contains definitions for secrets	.json
src	the source code of java implementation for automation process	
pom.xml	maven project - some configuration may be customized	
package.xml	describes the content of zip package - DO NOT EDIT THIS FILE	

Project package structure

Definitions of any entity must be provided in the form of jsons with the following structure:

Entity name	FQN	Structure

Automation Process	eu.ibagroup.easyrpa.cs.controller.dto.
	AutomationProcessDto
		{
		"capabilities": [
		"cap name"
		],
		"dedicated": true,
		"dedicatedNodes": [
		"node name"
		],
		"description": "This process automates a lot of work....",
		"input": {
		"executionPath": [
		"path"
		],
		"uuid": "id",
		"variables": {
		"additionalProp1": "string",
		"additionalProp2": "string",
		"additionalProp3": "string"
		}
		},
		"moduleClass": "eu.ibagroup.easyrpa.demo.
		FiveStepsDemoModule",
		"name": "Test Process",
		"params": [
		{
		"key": "string",
		"value": "string"
		}
		],
		"repositoryId": "eu.ibagroup:easy-rpa-sample:jar:full:
		0.1-SNAPSHOT",
		"repositoryType": "nexus"
		}

Datastore	eu.ibagroup.easyrpa.cs.controller.dto.
	DataStoreDto
		{
		  "description": "Sample datastore description",
		  "name": "Sample datastore"
		}

Secret	eu.ibagroup.easyrpa.cs.controller.dto.
	SecretVaultDto
		{
		  "alias": "string",
		  "value": "string"
		}

Package building

The following command will build the project and create a zip-package with all needed files including jar.

```
mvn clean package
```

The zip file of project will be generated and put into target directory

Package importing

There are two ways to import package into control server: from CLI with the help of in-build plugin and more conventional one - through UI

CLI

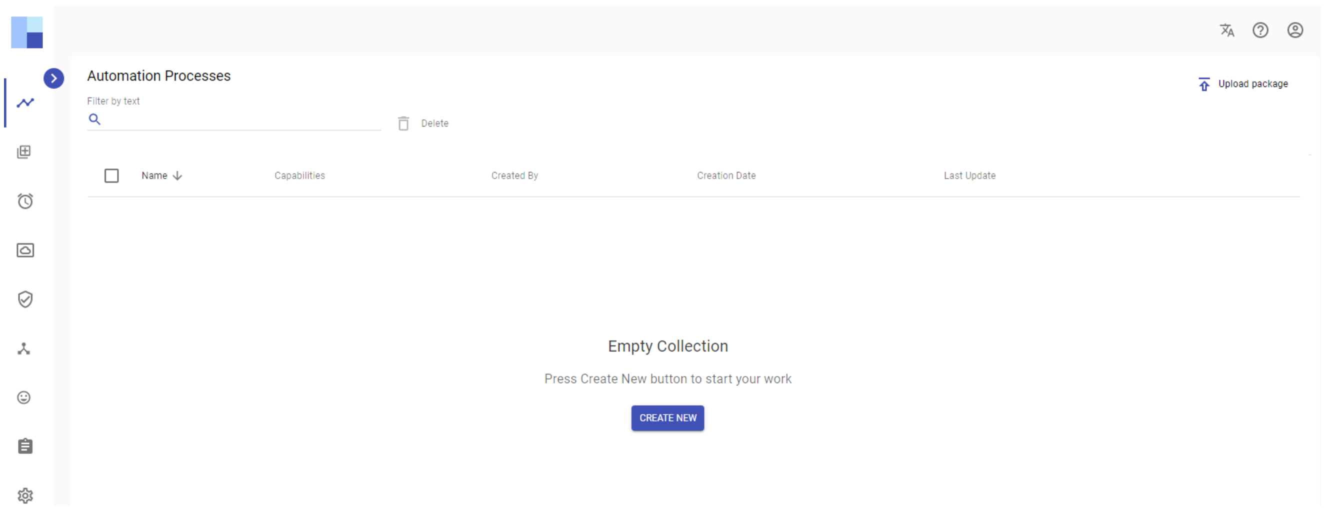
The following command will build the project, create a zip-package and upload it into CS

```
mvn clean package deploy -Pcs-upload
```

Note, if control server supports only SSL connections then plugin will use the trust store with location defined by pair of system variables node-agent.truststore.location/node-agent.truststore.password, overwise the default JDK trust store will be used

Note, that plugin is using OVERRIDE mode by default which results in creation of records which did not exist before and update of the existing ones.

UI



1. Navigate to page Automation Process Upload package
2. Choose the zip file to upload
3. Click Upload.

Control server will respond with report about operations it has done and their status. UI for package upload is using ADD mode by default which results in creation of records which did not exist before and fail in opposite case.

There are two possible scenarios:

1) There are no errors, and all components have been successfully created (no further actions required)

2) There are errors - some components failed due to conflicts with existing ones with the same name. In this case review the errors and choose different strategy to merge this components, such as OVERRIDE or SKIP. Press Upload button to perform upload operation once again.

Note, that the upload operation internally consists from two independent transactions - CS update and upload of jar file to Nexus. Upload to Nexus is not rolled back due to nexus limitations.

(v. 2.2) Work with Automation Process project

Using easyrpa launcher functionality

Making process launchable from Control Server

Create a master class that will contain the whole business process structure. Mark it with *ApModuleEntry* annotaion and make it extend *ApModuleBase* class, and implement *ApModule* class. Create *run()* method which should return *TaskOutput*. Add your process steps here (creation of the steps will be discussed further).

```
package com.example;
import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModule;
import eu.ibagroup.easyrpa.engine.apflow.ApModuleBase;
import eu.ibagroup.easyrpa.engine.apflow.TaskInput;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;

@ApModuleEntry(name = "Hello World")
public class ExampleProcess extends ApModuleBase implements ApModule {

    public TaskOutput run() throws Exception {
        return executeAsync(getInput(), ExampleStep.class).get();
    }
}
```

Launching process from IDE

When you develop a business process you would like to test it right in your IDE without the need of building a JAR. To make use of easyrpa launcher, dependency injection and other features in the debugging stage import ApModuleRunner and StandaloneConfigurationModule classes. Create main class and use aforementioned classes to execute run method in the same class.

```
    public static void main(String[] args) {
        ApModuleRunner runner = new ApModuleRunner();
        runner.launch(ExampleProcess.class, new TaskInput());
    }
```

Launch profiles

Refer the (v. 2.2) Quick Start Guide#Setupmodulerunconfigurations

---

Adding logic to your business process

Creating process step

Create a class extending *ApTaskBase* and implementing *ApTask.* Annotate it with *ApTaskEnt ry* annotation and add your task name and description as parameters. Your class should be looking something like that

```
package com.example;

import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.engine.apflow.ApTaskBase;

@ApTaskEntry(name = "Step 1", description = "Task that logs Hello World")
public class ExampleStep extends ApTaskBase implements ApTask {

}
```

Adding logic

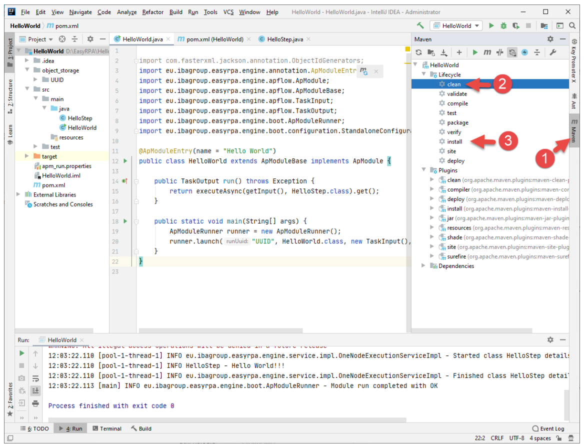
Now let's add logic here. Override execute() method. This is the most simple no input-output task we can build. It's made that way for the purpose of that demo.
Since we are using lombok, let's make use of it! Add *Slf4j* annotation to your class. This will give us access to logger right away. And let's log *"Hello World"* message in our *execute* method. Our class should be looking like this now

```
package com.example;

import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.engine.apflow.ApTaskBase;
import lombok.extern.slf4j.Slf4j;


@ApTaskEntry(name = "Step 1", description = "Task that logs Hello World")
@Slf4j
public class ExampleStep extends ApTaskBase implements ApTask {
    @Override
    public void execute() {
        log.info("Hello World!");
    }
}
```

Build Maven

You should do next steps to install Maven dependencies and build Maven:

1. Go to the Maven view (on the right), select your project and expand Lifecycle under it
2. Run clean by double click
3. Run install



```java
import com.fasterxml.jackson.annotation.ObjectIdGenerators;
import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntr
import eu.ibagroup.easyrpa.engine.apflow.ApModule;
import eu.ibagroup.easyrpa.engine.apflow.ApModuleBase;
import eu.ibagroup.easyrpa.engine.apflow.TaskInput;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import eu.ibagroup.easyrpa.engine.boot.ApModuleRunner;
import eu.ibagroup.easyrpa.engine.boot.configuration.StandaloneConfigura

@ApModuleEntry(name = "Hello World")
public class HelloWorld extends ApModuleBase implements ApModule {

    public TaskOutput run() throws Exception {
        return executeAsync(getInput(), HelloStep.class).get();
    }

    public static void main(String[] args) {
        ApModuleRunner runner = new ApModuleRunner();
        runner.launch( runUuid: "UUID", HelloWorld.class, new TaskInput(),
    }
}
```

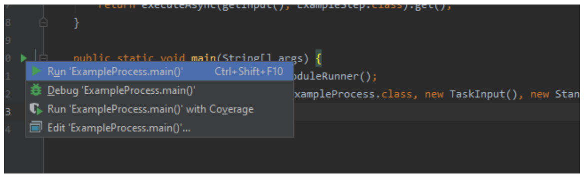
```
12:03:22.110 [pool-1-thread-1] INFO eu.ibagroup.easyrpa.engine.service.impl.OneNodeExecutionServiceImpl - Started class HelloStep detail
12:03:22.110 [pool-1-thread-1] INFO HelloStep - Hello World!!!
12:03:22.110 [pool-1-thread-1] INFO eu.ibagroup.easyrpa.engine.service.impl.OneNodeExecutionServiceImpl - Finished class HelloStep detai
12:03:22.113 [main] INFO eu.ibagroup.easyrpa.engine.boot.ApModuleRunner - Module run completed with OK

Process finished with exit code 0
```

Launching your process

⚠ Don't forget to add *apm_run.properties* file to your working directory!

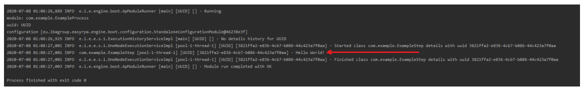
After launching your process for the first time, run configuration will be created. You can configure it by going to *RunEdit Configurations.* You can change your working directory there.

Невозможно предоставить {include} Включенная страница не может быть найдена.


Run 'ExampleProcess.main()'   Ctrl+Shift+F10
Debug 'ExampleProcess.main()'
Run 'ExampleProcess.main()' with Coverage
Edit 'ExampleProcess.main()'...

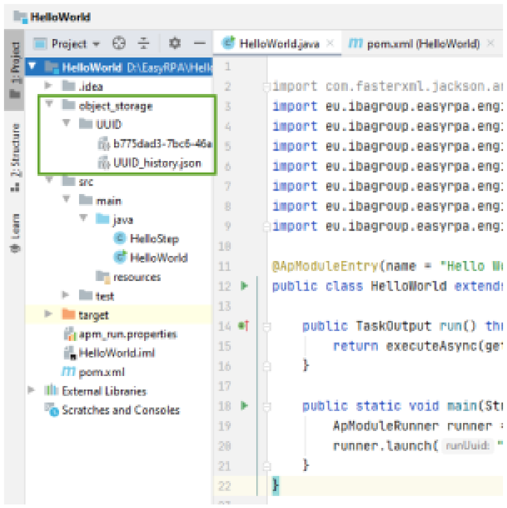
```java
    }

    public static void main(String[] args) {
        ...oduleRunner();
        ...xampleProcess.class, new TaskInput(), new Stan
```

If everything is correct you should be getting "Hello World" message printed out in console window.



```
2020-07-08 01:08:26,889 INFO  e.i.e.engine.boot.ApModuleRunner [main] [UUID] [] - Running
module: com.example.ExampleProcess
uuid: UUID
configuration [eu.ibagroup.easyrpa.engine.boot.configuration.StandaloneConfigurationModule@46238e3f]
2020-07-08 01:08:26,925 INFO  e.i.e.e.s.i.ExecutionHistoryServiceImpl [main] [UUID] [] - No details history for UUID
2020-07-08 01:08:27,001 INFO  e.i.e.e.s.i.OneNodeExecutionServiceImpl [pool-1-thread-1] [UUID] [3821ffa2-e836-4cb7-b086-44c423a7f0aa] - Started class com.example.ExampleStep details with uuid 3821ffa2-e836-4cb7-b086-44c423a7f0aa
2020-07-08 01:08:27,001 INFO  com.example.ExampleStep [pool-1-thread-1] [UUID] [3821ffa2-e836-4cb7-b086-44c423a7f0aa] - Hello World!
2020-07-08 01:08:27,001 INFO  e.i.e.e.s.i.OneNodeExecutionServiceImpl [pool-1-thread-1] [UUID] [3821ffa2-e836-4cb7-b086-44c423a7f0aa] - Finished class com.example.ExampleStep details with uuid 3821ffa2-e836-4cb7-b086-44c423a7f0aa
2020-07-08 01:08:27,003 INFO  e.i.e.engine.boot.ApModuleRunner [main] [UUID] [] - Module run completed with OK

Process finished with exit code 0
```

The *object_store* folder will created under you project structure after first run. UUID subfolder in the object_store contains the history of the ExampleProcess run, where "UUID" is process id, specified in the *runner.launch()* method. EasyRPA will check if this "UUID" folder is exist before run and will skip process run if folder is found. You should delete this folder to run ExampleProcess process again.



```
HelloWorld D:\EasyRPA\Hell
  .idea
  object_storage
    UUID
      b775dad3-7bc6-46a
      UUID_history.json
  src
    main
      java
        HelloStep
        HelloWorld
      resources
    test
  target
  apm_run.properties
  HelloWorld.iml
  pom.xml
External Libraries
Scratches and Consoles
```

```java

import com.fasterxml.jackson.a
import eu.ibagroup.easyrpa.eng
import eu.ibagroup.easyrpa.eng
import eu.ibagroup.easyrpa.eng
import eu.ibagroup.easyrpa.eng
import eu.ibagroup.easyrpa.eng
import eu.ibagroup.easyrpa.eng
import eu.ibagroup.easyrpa.eng

@ApModuleEntry(name = "Hello W
public class HelloWorld extend

    public TaskOutput run() th
        return executeAsync(ge
    }

    public static void main(St
        ApModuleRunner runner
        runner.launch( runUuid: "
    }
}
```

You can change the id string "UUID" to the random generated value by *UUID. randomUUID().toString()* to avoid skipping of class HelloStep.

(v. 2.2) Automation Process Framework

This section describes engine components which automation process can be built upon.

(v. 2.2) Task Execution API

Each business process you're automating contains one or more tasks. **Task** - is a special module, it's usually atomic business-operation such as "Read mails from mailbox" or "Save file to Data Base". Logic of your specific business-process should be described inside the special module which is called **AP** (Automation Process). **AP** is manipulating with Tasks and their input/outputs, to route the process to the correct flow.

AP (Automation Process)

AP module - is the special class which extends **ApModule**. The entry point of your AP class is method "**run**", which contains the high level logic of your business process, manipulates the tasks and their input/output and describes the process execution flow.

The AP class must be also annotated by **@ApModuleEntry** to provide the name of your AP, which will be displayed on Control Server.

Below there is an example of what AP class looks like:

MyDemoAp.java

```
import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModule;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import lombok.extern.slf4j.Slf4j;

@Slf4j
@ApModuleEntry(name = "My First Demo AP")
public class MyDemoAp extends ApModule {

    public TaskOutput run(){
        //automation process logic here
    }

}
```

Task

Task module - is the special class which extends **ApTask**. The entry point of your Task class is method "**execute**". Inside this method you have the access to task input and task output objects. Usually task contains some atomic business operation logic, like "Read mails from mailbox" or "Save file to Data Base". Also inside the task you can instantiate the driver object to work with some application which requires UI manipulations.

The Task classs must be also annotated by **@ApTaskEntry** to provide the name of your task which will be displayed on Control Server.

Example of Task class:

MyFirstTask.java

```
import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import lombok.extern.slf4j.Slf4j;

@ApTaskEntry(name = "My First Task")
@Slf4j
public class MyFirstTask extends ApTask {

    @Override
    public void execute() {
        //task logic here
    }

}
```

Local runner

To test and run your **AP** class locally, you need to create a **runner** and provide an initial input.

Below there is an example of class with static method "**main**" which is the entry point to run your AP:

LocalRunner.java

```
import eu.ibagroup.easyrpa.engine.boot.ApModuleRunner;
import lombok.extern.slf4j.Slf4j;

import java.util.UUID;

@Slf4j
public class LocalRunner {

        public static void main(String[] args) {
                ApModuleRunner.localLaunch(ExecuteAllAsyncDemoAp.class);
        }
}
```

Task execution

In the following examples we will use **four tasks** to demonstrates different ways of tasks execution. They're pretty the same, the only difference is that the first task sleeps longer that 2nd, 2nd sleeps longer than 3rd and 3rd sleeps longer than 4th:

Task1.java

```
import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import lombok.extern.slf4j.Slf4j;


@ApTaskEntry(name = "Task 1")
@Slf4j
public class Task1 extends ApTask {

    private static final int SLEEP_MILLIS = 4000;

    @Override
    public void execute() throws InterruptedException {
        Thread.sleep(SLEEP_MILLIS);
        log.info("Task 1 finished. Provided input: {}. It was sleeping {} millis", getInput().get("test_var"),
SLEEP_MILLIS);
        getOutput().put("test_var", "Output from Task1");
    }
}
```

Task2.java

```
import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import lombok.extern.slf4j.Slf4j;


@ApTaskEntry(name = "Task 2")
@Slf4j
public class Task2 extends ApTask {

    private static final int SLEEP_MILLIS = 3000;

    @Override
    public void execute() throws InterruptedException {
        Thread.sleep(SLEEP_MILLIS);
        log.info("Task 2 finished. Provided input: {}. It was sleeping {} millis", getInput().get("test_var"),
SLEEP_MILLIS);
        getOutput().put("test_var", "Output from Task2");
    }
}
```

Task3.java

```
import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import lombok.extern.slf4j.Slf4j;


@ApTaskEntry(name = "Task 3")
@Slf4j
public class Task3 extends ApTask {

    private static final int SLEEP_MILLIS = 2000;

    @Override
    public void execute() throws InterruptedException {
        Thread.sleep(SLEEP_MILLIS);
        log.info("Task 3 finished. Provided input: {}. It was sleeping {} millis", getInput().get("test_var"),
SLEEP_MILLIS);
        getOutput().put("test_var", "Output from Task3");
    }
}
```

Task4.java

```
import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import lombok.extern.slf4j.Slf4j;


@ApTaskEntry(name = "Task 4")
@Slf4j
public class Task4 extends ApTask {

    private static final int SLEEP_MILLIS = 1000;

    @Override
    public void execute() throws InterruptedException {
        Thread.sleep(SLEEP_MILLIS);
        log.info("Task 4 finished. Provided input: {}. It was sleeping {} millis", getInput().get("test_var"),
SLEEP_MILLIS);
        getOutput().put("test_var", "Output from Task4");
    }
}
```

execute

All **execute** methods will start execution of your tasks asynchronously, so task execution will be started, but the main thread of your AP will not wait for result and will continue processing the next line of code. It's useful when you don't need the output results immediately from that task and your AP can continue performing the following instructions.

Instead of **TaskOutput** this method returns **CompletableFuture<TaskOutput>** object. To get the output from that object, you need to call method "**get()**". Method "get()" **will return** you the **TaskOutput immediately** if **task has already been completed**. And it **will block your main AP thread** and **will wait until task is completed** in case if **task hasn't been completed yet**.

execute

```
CompletableFuture<TaskOutput> execute(TaskInput input, Class<? extends ApTask>... task)
CompletableFuture<TaskOutput> execute(TaskInput input, ExecutionService.MergeStrategy mergeStrategy, Class<?
extends ApTask>... task)
CompletableFuture<TaskOutput> execute(TaskOutput output, Class<? extends ApTask>... task)
CompletableFuture<TaskOutput> execute(TaskOutput prevTaskOutput, ExecutionService.MergeStrategy mergeStrategy,
Class<? extends ApTask>... classes)
Function<TaskOutput, CompletableFuture<TaskOutput>> execute(Class<? extends ApTask>... classes)
Function<TaskOutput, CompletableFuture<TaskOutput>> execute(ExecutionService.MergeStrategy mergeStrategy,
Class<? extends ApTask>... classes)
```

- **input** - your input object which came from local runner in case of local execution or input object which is provided on Control Server in case of executiomergeStrategyn on the server.
- **task**- class of the task your want to execute, e.g. "**MyFirstTask.class**".
- **mergeStrategy** - BiConsumer object (usually lambda expression) where you can define your own merge strategy.
- **output** - **TaskOutput** object which allows you to use the output from previously executed task as an input for the next task.

Synchronous Task Execution

This is a simple method to execute your single task with some provided input.

execute

```
CompletableFuture<TaskOutput> execute(TaskInput input, Class<? extends ApTask>... task)
```

Where:

- **input** - your input object which came from local runner in case of local execution or input object which is provided on Control Server in case of executiomergeStrategyn on the server.
- **task**- class of the task your want to execute, e.g. "**MyFirstTask.class**".

To get the output from that object, you need to call method "**get()**"

Also instead of **TaskInput** object this method may accept **TaskOutput** object which allows you to use the output from previously executed task as an input for the next task:

```
CompletableFuture execute(TaskOutput prevTaskOutput, Class<? extends ApTask>... task)
```

Below you'll find the complete example of AP which executes four tasks where all the outputs are used as the inputs for the following tasks:

ExecuteDemoAp.java

```
package eu.ibagroup.easyrpa.taskexecution;

import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModule;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import eu.ibagroup.easyrpa.engine.boot.ApModuleRunner;
import eu.ibagroup.easyrpa.taskexecution.task.Task1;
import eu.ibagroup.easyrpa.taskexecution.task.Task2;
import eu.ibagroup.easyrpa.taskexecution.task.Task3;
import eu.ibagroup.easyrpa.taskexecution.task.Task4;
import lombok.extern.slf4j.Slf4j;

@Slf4j
@ApModuleEntry(name = "Task Execution Demo")
public class ExecuteDemoAp extends ApModule {

        public static void main(String[] args) {
                ApModuleRunner.localLaunch(ExecuteDemoAp.class);
        }

        public TaskOutput run() throws Exception {
                log.info("execute method test started");
                TaskOutput previousOutput = execute(getInput(), Task1.class).get();
                previousOutput = execute(previousOutput, Task2.class).get();
                previousOutput = execute(previousOutput, Task3.class).get();
                previousOutput = execute(previousOutput, Task4.class).get();

                log.info("execute method test finished. Final output: {}", previousOutput.get("test_var"));
                return previousOutput;
        }
}
```

After execution is finished we will see the following logs:

```
2020-08-26 19:15:28,740 INFO  e.i.e.taskexecution.ExecuteDemoAp [main] - execute method test started
2020-08-26 19:15:32,805 INFO  e.i.easyrpa.taskexecution.task.Task1 [main] - Task 1 finished. Provided input:
null. It was sleeping 4000 millis
2020-08-26 19:15:35,822 INFO  e.i.easyrpa.taskexecution.task.Task2 [main] - Task 2 finished. Provided input:
Output from Task1. It was sleeping 3000 millis
2020-08-26 19:15:37,838 INFO  e.i.easyrpa.taskexecution.task.Task3 [main] - Task 3 finished. Provided input:
Output from Task2. It was sleeping 2000 millis
2020-08-26 19:15:38,860 INFO  e.i.easyrpa.taskexecution.task.Task4 [main] - Task 4 finished. Provided input:
Output from Task3. It was sleeping 1000 millis
2020-08-26 19:15:38,866 INFO  e.i.e.taskexecution.ExecuteDemoAp [main] - execute method test finished. Final
output: Output from Task4
```

Asynchronous Task Execution

Below you'll find the complete example of AP which executes four tasks asynchronously and prints the output from all of them separately:

ExecuteAsyncDemoAp.java

```
package eu.ibagroup.easyrpa.taskexecution;

import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModule;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import eu.ibagroup.easyrpa.engine.boot.ApModuleRunner;
import eu.ibagroup.easyrpa.taskexecution.task.Task1;
import eu.ibagroup.easyrpa.taskexecution.task.Task2;
import eu.ibagroup.easyrpa.taskexecution.task.Task3;
import eu.ibagroup.easyrpa.taskexecution.task.Task4;
import lombok.extern.slf4j.Slf4j;

import java.util.concurrent.CompletableFuture;

@Slf4j
@ApModuleEntry(name = "Execute Async Demo")
public class ExecuteAsyncDemoAp extends ApModule {

        public static void main(String[] args) {
                ApModuleRunner.localLaunch(ExecuteAsyncDemoAp.class);
        }

        public TaskOutput run() throws Exception {
                log.info("executeAsync method test started");

                CompletableFuture<TaskOutput> completableFuture1 = execute(getInput(), Task1.class);
                CompletableFuture<TaskOutput> completableFuture2 = execute(getInput(), Task2.class);
                CompletableFuture<TaskOutput> completableFuture3 = execute(getInput(), Task3.class);
                CompletableFuture<TaskOutput> completableFuture4 = execute(getInput(), Task4.class);

                log.info("executeAsync method test finished. Final output Task1: {}; Task2: {}; Task3: {};
Task4: {}", completableFuture1.get().get("test_var"),
                completableFuture2.get().get("test_var"), completableFuture3.get().get("test_var"),
completableFuture4.get().get("test_var"));

                return completableFuture4.get();
        }
}
```

After the execution is finished we will see the following logs:

```
2020-08-26 19:37:55,503 INFO e.i.e.t.ExecuteAsyncDemoAp [main] - executeAsync method test started
2020-08-26 19:37:56,574 INFO  e.i.easyrpa.taskexecution.task.Task4 [pool-1-thread-4] - Task 4 finished.
Provided input: null. It was sleeping 1000 millis
2020-08-26 19:37:57,574 INFO  e.i.easyrpa.taskexecution.task.Task3 [pool-1-thread-3] - Task 3 finished.
Provided input: Output from Task4. It was sleeping 2000 millis
2020-08-26 19:37:58,574 INFO  e.i.easyrpa.taskexecution.task.Task2 [pool-1-thread-2] - Task 2 finished.
Provided input: Output from Task3. It was sleeping 3000 millis
2020-08-26 19:37:59,573 INFO  e.i.easyrpa.taskexecution.task.Task1 [pool-1-thread-1] - Task 1 finished.
Provided input: Output from Task2. It was sleeping 4000 millis
2020-08-26 19:37:59,578 INFO e.i.e.t.ExecuteAsyncDemoAp [main] - executeAsync method test finished. Final
output Task1: Output from Task1; Task2: Output from Task2; Task3: Output from Task3; Task4: Output from Task4
```

Additionally **CompletableFuture** API provides possibility to combine executions in chains, where output from previous task will be used as an input for the next task automatically. For that you can call the method "**thenCompose**" from CompletableFuture object and provide the "**executeAsync**" methods, which returns the **java.util.function.Function** object. For that you should use the following methods signatures:

```
Function<TaskOutput, CompletableFuture<TaskOutput>> execute(Class<? extends ApTask>... classes)
```

As you may see, the only different here is that those methods don't accept the input as argument, because it will be provided automatically.

So the example of using it is the following:

```
 CompletableFuture<TaskOutput> finalCompletableFuture = execute(getInput(), Task1.class)
               .thenCompose(execute(Task2.class))
               .thenCompose(execute(Task3.class))
               .thenCompose(execute(Task4.class));
```

Below you'll find the complete example of AP which executes four tasks in the chain, where the output from previous task is used as an input for the next task automatically. In such case behavior of the execution will be the same as in "**execute**" method, where you provide inputs/outputs chain manually:

ExecuteAsyncChainDemoAp.java

```
@Slf4j
@ApModuleEntry(name = "Execute Async Chain Demo")
public class ExecuteAsyncChainDemoAp extends ApModule {

        public static void main(String[] args) {
                ApModuleRunner.localLaunch(ExecuteAsyncChainDemoAp.class);
        }

        public TaskOutput run() throws Exception {
                log.error("executeAsync method chain test started");

                CompletableFuture<TaskOutput> finalCompletableFuture = execute(getInput(), Task1.class)
                .thenCompose(execute(Task2.class))
                .thenCompose(execute(Task3.class))
                .thenCompose(execute(Task4.class));

                TaskOutput finalOutput = finalCompletableFuture.get();

                log.error("executeAsync method chain test finished. Final output: {}; ", finalOutput.get
("test_var"));

                return finalOutput;
        }
}
```

After the execution is finished will see the following logs:

```
2020-08-26 19:51:07,212 INFO e.i.e.t.ExecuteAsyncChainDemoAp [main] - executeAsync method chain test started
2020-08-26 19:51:11,268 INFO  e.i.easyrpa.taskexecution.task.Task1 [pool-1-thread-1] - Task 1 finished.
Provided input: null. It was sleeping 4000 millis
2020-08-26 19:51:14,284 INFO  e.i.easyrpa.taskexecution.task.Task2 [pool-1-thread-2] - Task 2 finished.
Provided input: Output from Task1. It was sleeping 3000 millis
2020-08-26 19:51:16,298 INFO  e.i.easyrpa.taskexecution.task.Task3 [pool-1-thread-1] - Task 3 finished.
Provided input: Output from Task2. It was sleeping 2000 millis
2020-08-26 19:51:17,312 INFO  e.i.easyrpa.taskexecution.task.Task4 [pool-1-thread-2] - Task 4 finished.
Provided input: Output from Task3. It was sleeping 1000 millis
2020-08-26 19:51:17,317 INFO e.i.e.t.ExecuteAsyncChainDemoAp [main] - executeAsync method chain test finished.
Final output: Output from Task4;
```

Asynchronous Set of Tasks Execution

You can use this method to trigger the asynchronous execution of more than one task with the same input. It has the following method signature:

```
CompletableFuture<TaskOutput> execute(TaskOutput output, Class<? extends ApTask>... task)
```

Where:

- **input** - your input object which came from local runner in case of local execution or input object which is provided on Control Server in case of execution on the server.
- **task**- classes of the tasks your want to execute, e.g. "**MyFirstTask.class, MySecondTask.class,**".

Returns: **CompletableFuture** object on which you can call "**get()**" method to get **TaskOutput** object.

So the example of using it:

```
CompletableFuture<TaskOutput> completableFutureCombined = execute(getInput(), Task1.class, Task2.class, Task3.
class, Task4.class);
```

The **final output** will be the result of **merging of all outputs**. By default there is the merge strategy that in case of conflict (output variable with the same name) - last task provided for that method will overwrite such duplicated variable.

Let's see the following example of the complete AP:

ExecuteAllAsyncDemoAp.java

```
import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModuleBase;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import eu.ibagroup.easyrpa.taskexecution.task.Task1;
import eu.ibagroup.easyrpa.taskexecution.task.Task2;
import eu.ibagroup.easyrpa.taskexecution.task.Task3;
import eu.ibagroup.easyrpa.taskexecution.task.Task4;
import lombok.extern.slf4j.Slf4j;

import java.util.concurrent.CompletableFuture;

@Slf4j
@ApModuleEntry(name = "Execute All Async Demo")
public class ExecuteAllAsyncDemoAp extends ApModuleBase {

    public TaskOutput run() throws Exception {
        log.info("executeSubtasks method test started");
        CompletableFuture<TaskOutput> completableFutureCombined = executeSubtasks(getInput(), Task1.class,
Task2.class, Task3.class, Task4.class);
        TaskOutput output = completableFutureCombined.get();
        log.info("executeSubtasks method test finished. Final output: {};", output.get("test_var"));
        return output;
    }

}
```

After the execution is finished we will see the following logs:

```
2020-08-26 20:07:43,537 INFO  e.i.e.t.ExecuteAllAsyncDemoAp [main] - executeSubtasks method test started
2020-08-26 20:07:44,644 INFO  e.i.easyrpa.taskexecution.task.Task4 [pool-1-thread-5] - Task 4 finished.
Provided input: null. It was sleeping 1000 millis
2020-08-26 20:07:45,634 INFO  e.i.easyrpa.taskexecution.task.Task3 [pool-1-thread-4] - Task 3 finished.
Provided input: null. It was sleeping 2000 millis
2020-08-26 20:07:46,638 INFO  e.i.easyrpa.taskexecution.task.Task2 [pool-1-thread-3] - Task 2 finished.
Provided input: null. It was sleeping 3000 millis
2020-08-26 20:07:47,627 INFO  e.i.easyrpa.taskexecution.task.Task1 [pool-1-thread-2] - Task 1 finished.
Provided input: null. It was sleeping 4000 millis
2020-08-26 20:07:47,636 INFO  e.i.e.t.ExecuteAllAsyncDemoAp [main] - executeSubtasks method test finished.
Final output: Output from Task4;
```

It's possible to define **your own output merge strategy**. For that you should use the following method sugnature:

```
CompletableFuture<TaskOutput> execute(TaskInput input, ExecutionService.MergeStrategy mergeStrategy, Class<?
extends ApTask>... task)
```

Where:

- **input** - your input object which came from local runner in case of local execution or input object which is provided on Control Server in case of execution on the server.
- **mergeStrategy** - BiConsumer object (usually lambda expression) where you can define your own merge strategy.
- **classes**- classes of the tasks your want to execute, e.g. "**MyFirstTask.class, MySecondTask.class,**".

Returns: **CompletableFuture** object on which you can call "**get()**" method to get **TaskOutput** object.

Let's see the following example where we define our own merge strategy. In case of conflicts (output variable with the same name) - it combines the value using the pipe ("|") symbol:

ExecuteAllAsyncWithMergeStrategyDemoAp.java

```
package eu.ibagroup.easyrpa.taskexecution;

import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModule;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import eu.ibagroup.easyrpa.engine.boot.ApModuleRunner;
import eu.ibagroup.easyrpa.engine.service.ExecutionService;
import eu.ibagroup.easyrpa.taskexecution.task.Task1;
import eu.ibagroup.easyrpa.taskexecution.task.Task2;
import eu.ibagroup.easyrpa.taskexecution.task.Task3;
import eu.ibagroup.easyrpa.taskexecution.task.Task4;
import lombok.extern.slf4j.Slf4j;

import java.util.concurrent.CompletableFuture;

@Slf4j
@ApModuleEntry(name = "Execute All Async with merge strategy Demo")
public class ExecuteAllAsyncWithMergeStrategyDemoAp extends ApModule {

        public static void main(String[] args) {
                ApModuleRunner.localLaunch(ExecuteAllAsyncWithMergeStrategyDemoAp.class);
        }

        public TaskOutput run() throws Exception {
                log.info("executeAllAsync with merge strategy method test started");

                ExecutionService.MergeStrategy combineWithPipeMergeStrategy = (finalOutput,
finishedTaskVariables) -> {
                        for (String key : finishedTaskVariables.keySet()) {
                                if (finalOutput.containsKey(key)) {
                                        String currentFinalValue = finalOutput.get(key);
                                        currentFinalValue = currentFinalValue + "|" + finishedTaskVariables.get
(key);
                                        finalOutput.replace(key, currentFinalValue);
                                } else {
                                        finalOutput.put(key, finishedTaskVariables.get(key));
                                }
                        }
                };

                CompletableFuture<TaskOutput> completableFutureCombined = execute(getInput(),
combineWithPipeMergeStrategy, Task1.class, Task2.class, Task3.class, Task4.class);
                TaskOutput output = completableFutureCombined.get();
                log.info("executeAllAsync with merge strategy method test finished. Final output: {};", output.
get("test_var"));
                return output;
        }
}
```

After the execution we will see the following logs:

```
2020-08-26 20:17:53,658 INFO  e.i.e.t.ExecuteAllAsyncWithMergeStrategyDemoAp [main] - executeSubtasks with
merge strategy method test started
2020-08-26 20:17:54,747 INFO  e.i.easyrpa.taskexecution.task.Task4 [pool-1-thread-5] - Task 4 finished.
Provided input: null. It was sleeping 1000 millis
2020-08-26 20:17:55,753 INFO  e.i.easyrpa.taskexecution.task.Task3 [pool-1-thread-4] - Task 3 finished.
```

```
Provided input: null. It was sleeping 2000 millis
2020-08-26 20:17:56,758 INFO  e.i.easyrpa.taskexecution.task.Task2 [pool-1-thread-3] - Task 2 finished.
Provided input: null. It was sleeping 3000 millis
2020-08-26 20:17:57,743 INFO  e.i.easyrpa.taskexecution.task.Task1 [pool-1-thread-2] - Task 1 finished.
Provided input: null. It was sleeping 4000 millis
2020-08-26 20:17:57,754 INFO  e.i.e.t.ExecuteAllAsyncWithMergeStrategyDemoAp [main] - executeSubtasks with
merge strategy method test finished. Final output: Output from Task1|Output from Task2|Output from Task3|Output
from Task4;
```

(v. 2.2) Input/Output usage

What is Input/Output data?

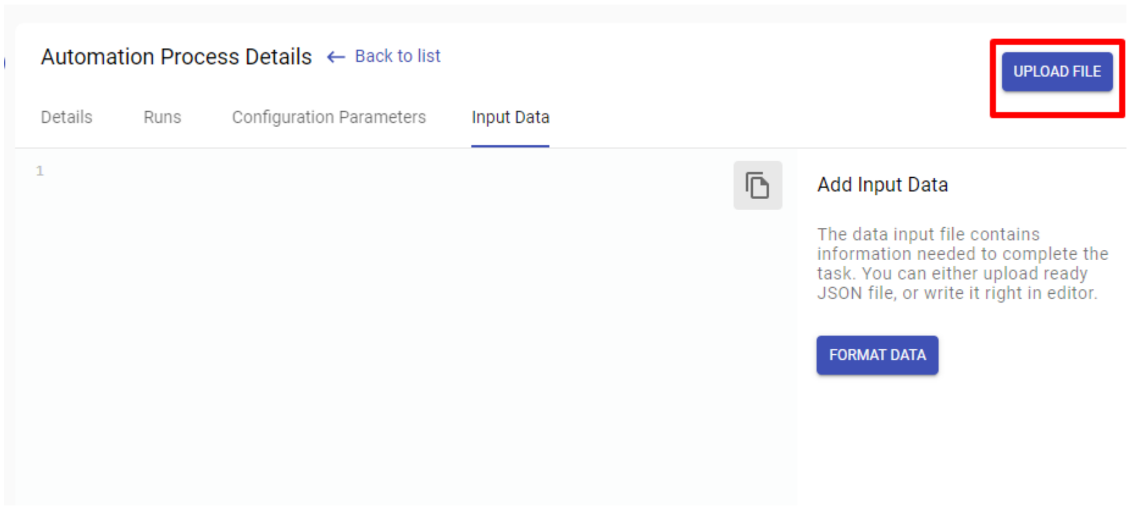
Input and Output objects are necessary to transfer data to the process tasks and get the result. Most often, they are used to transfer data identifiers for processing, data transfer between steps (usually, the data is stored in the database, and only the keys are sent to the next step), and to get the output data - the results of work. It can be business data or information about successful execution.

Input/Output objects structure

You can save any serializable object as an input or output object. Input/output objects store data in the variable *HashMap<String, String> variables*. The key is the variable name, and the value is the data stored in json format. When reading/saving data in input /output, the added object is automatically serialized/deserialized into json string.

Usage

To upload data to the process input, go to Automation Process Details -> Input Data and upload a data file in json format.



The data input file contains information needed to complete the task. You can either upload ready JSON file, or write it right in editor.

After that, to use the data uploaded through the control server or created in the code, you need to declare a variable with the *@Input* annotation and the corresponding name and type.

```
public class SampleTask extends ApTask {

    @Input(key = "newKey")
    private String myInputString;

        @Input
        private MyTransferClass myTransferObject;
...
```

In the example above, we expect 2 variables as the input.
The first variable is of the *String* type. *@Input(key = "newKey")* means that the variable was passed to the process with the *newKey*

key.
The second variable of *MyTransferClass* type was passed with *myTransferObject* key without any additional options. So we just declare the variable with the corresponding type and name, specify the *@Input* annotation and then we can use the data.

To send data to the output, the *@Output* annotation is used when declaring a variable. If you want to get the data, make changes and pass it on, you can use both annotations at the same time.

```
public class SampleTask extends ApTask {

        @Output
        @Input
        private MyTransferClass myTransferObject;

    @Output
    private String myOutputString;
...
```

In this example we get the input object *myTransferObject* and pass it to the output. We also create an additional variable *myOutputString* which is also passed to the next steps.

Example

In this example, we work with the Book class. The example consists of two steps. In the first step, we receive the data and pass it to the second step. The second step takes the data from the first step, makes changes and sends it to the output.

Class Book is the entity we'll save in the DataStore and then pass between steps.

Book.java

```
import eu.ibagroup.easyrpa.persistence.annotations.Column;
import eu.ibagroup.easyrpa.persistence.annotations.Entity;
import eu.ibagroup.easyrpa.persistence.annotations.Id;
import lombok.Data;
import lombok.NoArgsConstructor;

@Entity(value = "io_example_books")
@NoArgsConstructor
@Data
public class Book {

    @Id
    @Column("index")
    private String index;

    @Column("name")
    private String name;

    @Column("author")
    private String author;

    @Column("count")
    private String count;

    public Book(String index, String name, String author) {
        this.index = index;
        this.name = name;
        this.author = author;
    }

    public Book(String index, String name, String author, String count) {
        this.index = index;
        this.name = name;
        this.author = author;
        this.count = count;
    }
}
```

Transferring the entire object may require a lot of resources. Therefore, the right approach is to save objects in the DataStore and transfer their identifiers between the steps.

For information transfer, a special DTO object is created which contains the process identifier and the transferred data - the entity identifiers in the DataStore.

Book.java

```
import lombok.Getter;
import lombok.ToString;

import java.util.ArrayList;
import java.util.List;

@ToString
public class BooksResultDto {

    @Getter
    private final String uuid;

    @Getter
    private final List<String> bookIds;

    public BooksResultDto(String uuid) {
        this.bookIds = new ArrayList<>();
        this.uuid = uuid;
    }

    public void add(String id) {
        bookIds.add(id);
    }
}
```

In this example, there are enough methods available by default for working with DataStore, so the BookRepository interface remains empty.

BookRepository.java

```
import eu.ibagroup.easyrpa.io.entities.Book;
import eu.ibagroup.easyrpa.persistence.CrudRepository;

import java.util.ArrayList;
import java.util.List;

public interface BookRepository extends CrudRepository<Book, String> {
}
```

SampleTaskOne receives the books, saves them in the DataStore and passes the DTO object with identifiers to the next step.

SampleTaskOne.java

```
import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.annotation.Output;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.io.BookRepository;
import eu.ibagroup.easyrpa.io.BooksResultDto;
import eu.ibagroup.easyrpa.io.entities.Book;
import lombok.extern.slf4j.Slf4j;

import javax.inject.Inject;
import java.util.Arrays;
import java.util.List;
import java.util.UUID;

@ApTaskEntry(name = "Sample Task One")
```

```
@Slf4j
public class SampleTaskOne extends ApTask {

    //key to output identification
    public static final String BOOK_KEY = "Book";

    @Output(key = BOOK_KEY)
    private BooksResultDto booksResult;

    @Inject
    private BookRepository bookRepository;

    @Override
    public void execute() {

        booksResult = new BooksResultDto(getUuid());

        for (Book book : getBooks()){
            //save to DataStore
            bookRepository.save(book);

            //add ID to output
            booksResult.add(book.getIndex());
        }
    }

    //Getting books. Data can be obtained from the client program, database, scraped from the website, etc.
    private List<Book> getBooks() {
        Book thinkingInJava = new Book(UUID.randomUUID().toString(), "Thinking in Java", "Bruce Eckel");
        Book cipollino = new Book(UUID.randomUUID().toString(), "Le avventure di Cipollino", "Giovanni
Francesco Rodari");
        Book warAndPeace = new Book(UUID.randomUUID().toString(), "War and Peace", "Lev Tolstoy");

        return Arrays.asList(thinkingInJava, cipollino, warAndPeace);
    }
}
```

SampleTaskTwo receives the input data from SampleTaskOne using the @Input annotation.
Then, using identifiers from the input DTO object, the data is extracted from the DataStore. After performing manipulations with the obtained data, it is updated in the DataStore, identifiers of new entities are added to the output-object

The annotation @Output indicates that after processing the data will be transferred to the next step.

SampleTaskTwo.java

```
import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.annotation.Input;
import eu.ibagroup.easyrpa.engine.annotation.Output;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.io.BookRepository;
import eu.ibagroup.easyrpa.io.BooksResultDto;
import eu.ibagroup.easyrpa.io.entities.Book;
import lombok.extern.slf4j.Slf4j;

import javax.inject.Inject;

import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import java.util.UUID;

import static eu.ibagroup.easyrpa.io.task.SampleTaskOne.BOOK_KEY;


@ApTaskEntry(name = "Sample Task Two")
@Slf4j
public class SampleTaskTwo extends ApTask {

    @Input(key = BOOK_KEY)
    @Output
    private BooksResultDto booksResult;

    @Inject
    private BookRepository bookRepository;

    @Override
    public void execute() {

        //get entities
        List<Book> books = new ArrayList<>();
        for (String id : booksResult.getBookIds()) {
            books.add(bookRepository.findById(id));
        }

        //update entities
        for (Book book : books) {
            book.setCount(String.valueOf(getBooksCount(book.getName())));
                        bookRepository.save(book);
        }

        //add new entities to repository
        Book newBook = getNewBook();
        bookRepository.save(newBook);

        //add new entities to result
        booksResult.add(newBook.getIndex());
    }

    private Book getNewBook() {
        return new Book(UUID.randomUUID().toString(), "The Godfather", "Mario Puzo", "50");
    }

    //Getting additional information. Data can be obtained from the client program, database, scraped from the
website, etc.
    private int getBooksCount(String name) {
        return new Random().nextInt(100);
    }
}
```

The automation process to run tasks:

SampleTask.java

```
import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModule;
import eu.ibagroup.easyrpa.engine.apflow.TaskInput;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import eu.ibagroup.easyrpa.engine.boot.ApModuleRunner;
import eu.ibagroup.easyrpa.engine.boot.ConfigurationModule;
import eu.ibagroup.easyrpa.engine.boot.configuration.StandaloneConfigurationModule;
import eu.ibagroup.easyrpa.io.task.SampleTaskOne;
import eu.ibagroup.easyrpa.io.task.SampleTaskTwo;
import lombok.extern.slf4j.Slf4j;

import java.util.UUID;

@Slf4j
@ApModuleEntry(name = "Sample Automation Process of working with task input/output")
public class IOSampleAp extends ApModule {
    public TaskOutput run() throws Exception {
        return executeAsync(getInput(), SampleTaskOne.class)
                .thenCompose(executeAsync(SampleTaskTwo.class))
                .get();
    }

    public static void main(String[] args) {
        ApModuleRunner runner = new ApModuleRunner();
        String uuid = UUID.randomUUID().toString();
        ConfigurationModule configurationModule = new StandaloneConfigurationModule();

        runner.launch(uuid, IOSampleAp.class, new TaskInput(), configurationModule);
    }
}
```

(v. 2.2) Annotations

Contents

AP components

These annotations are mandatory to successfully run AP. Otherwise the code will break apart and won't be started

@ApModuleEntry

This one signifies that target class represents Automation Process

When node executes AP from jar package the class annotated by ApModuleEntry will be started

In case there is more than one annotated class the one with `defaultRunModule=true` will be started

Annotated class is normally extended from `ApModuleBase` class so its execution starts from `run()` method

```
@ApModuleEntry(defaultRunModule=true)
public class RdpAp extends ApModuleBase {
    @Override
    public TaskOutput run() {
        // workflow resulting in TaskOutput
    }
}
```

@ApTaskEntry

ApTaskEntry can be viewed as a single step inside Automation Process

Annotated class is normally extended from ApTaskBase with `execute()` method as entry point, which encapsulates the business logic of this step

```
@ApTaskEntry(name = "Calculator task", description = "Demonstrates image-based automation")
public class PerformCalcMultiUser extends ApTaskBase {
    @Override
    public void execute() {
                // sequence of actions
    }
}
```

Task level annotations

These annotations can be applied to fields and methods inside AP task entry

@Output

The field annotated by @Output will be exported as a part of task output after its execution is finished.

It can be later imported into successive task using @Input

```
@ApTaskEntry(name = "Extract Debtors")
public class ExtractDebtors extends ApTaskBase {
    @Output(key = "Debtor")
    private DebtorsResultTo debtorsResult;

    @Override
    public void execute() {
        debtorsResult = new DebtorsResultTo("abc");
    }
}
```

@Input

@Input annotation allows import of the value exported from previous task

```
@ApTaskEntry(name = "Check Bank Guarantees")
public class CheckBankGuarantees extends ApTaskBase {

    @Input(key = "Debtor")
    private DebtorsResultTo debtorsResult;

    @Override
    public void execute() {
                // actions performed on debtorsResult
    }
}
```

both `@Input` and `@Output` have properties `key` and `required`

`key` - the key associated with stored and retrieved value

`required` - the boolean flag which enforces associated value to be checked on null

@Configuration

@Configuration retrieves value by name from configuration service or secret vault.

```
@Configuration(value = "invoiceplane.client.url", defaultValue = "http://10.224.0.35:8085/index.php/sessions
/login")
protected String invoicePlaneUrl;

@Configuration(value = "invoiceplane.secrets", defaultValue = "{\"user\": \"admin@ibagroup.eu\",\"password\": \"
o66Lc1Jn6Z\"}")
protected SecretCredentials invoicePlaneCredentials;

@Configuration(value = "products_count", defaultValue = "20")
private int maxProductsCountToRead;
```

@AfterInit

`@AfterInit` method will be called after base task is initialized and before `execute()` is called

It makes a good practice to initialize config settings and drivers in a such way

```
@ApTaskEntry(name = "Extract Debtors", description = "Extract debtors from 1C Accounting")
public class ExtractDebtors extends ApTaskBase {
    private BrowserDriver driver;
    private String url;

    @AfterInit
    public void init() {
        url = getConfigurationService().get("acc.client.url");
        driver = getDriver(BrowserDriver.class, new HashMap<Enum, Object>() {{
            put(DriverParams.SELENIUM_NODE_CAPABILITIES, new ChromeOptions());
        }});
    }

    @Override
    public void execute() {
        OnecApplication application = new OnecApplication(driver);
        LoginPage loginPage = application.open(url);
                // ...
    }
}
```

@OnError

Normally `execute()` method can propagate exceptions outside of the task entry

In order to capture such exceptions and separate exception-handling logic it is recommended to create a specific method annotated with @OnError

```
@ApTaskEntry(name = "Arithmetic Calculation")
public class ArithmeticCalculation extends ApTaskBase {
    @Override
    public void execute() throws Throwable {
        int a = 1/0;
    }

    @OnError
    public void onError(Throwable throwable) {
        System.err.println("Arithmetic error occured!");
    }
}
```

Page elements

This group annotates Page Object elements such as input fields, buttons, window panels etc

@FIndBy

Indicates the locating mechanism for page element. Can receive field id, name, xpath and other parameters depending on the driver. More on this in How to Locate Interface Elements

```
@FindBy(name = "One")
private RemoteWebElement one;

@FindBy(xpath = "//*[@id='userName']")
private RemoteWebElement username;
```

@Wait

Indicates wait timeout in seconds and wait function as Selenium FluentWait condition. Can be paired with @FindBy annotation

```
    @FindBy(xpath = "//input[@id=\"Name\"]")
    @Wait(waitFunc = Wait.WaitFunc.VISIBLE)
    private WebElement nameField;
    @FindBy(xpath = "//a[@id=\"searchButton\"]")
    @Wait(value = 30, waitFunc = Wait.WaitFunc.CLICKABLE)
    private WebElement searchBtn;
```

@Image

Marks the element to be detected by screen driver and receives image url relative to resources/images directory

```
    @Image(url = "navigation-btn-1.png")
    private IMElement navigation;
```

@WithTimeout

Indicates wait timeout for screen driver elements, receives time in seconds as parameter. Can be paired with @Image annotation.

```
    @Image(url = "windows-start-1.png")
    @WithTimeout(time = 3)
    private IMElement start;
```

Lombok Annotations

This set of annotations is provided by Lombok, the code generating library.

They help to avoid repetitive code writing and hence reduces the typical errors in daily tasks.

We covered Lombok use cases in separate article.

(v. 2.2) Exception Handling

EasyRPA Exception Handling Mechanism

Developer has three options when dealing with exception inside AP :

- catch it inside AP and hence prevent AP from termination with 'failed' status
- rethrow or left it propagated down the call stack to be eventually handled by engine
- declare `@OnError` handler to tell engine the way to deal with exceptions not handled directly by developer

The best way to rethrow is to define AP specific exceptions, more on this further

Define Specific AP Exceptions

Developer can define AP-related errors by implementing `eu.ibagroup.easyrpa.engine.exception.ErrorDetails` interface onto enumeration class.

It is recommended to store error messages as resource bundle.

InvoicePlaneError.java

```
public enum InvoicePlaneError implements ErrorDetails {

    LOGIN_FAILED,
    ADD_PRODUCT_FAILED;

    private static ResourceBundle rb = ResourceBundle.getBundle("invoiceplane_error");

    public String getMessageTemplate() {
        return rb.getString(this.name());
    }
}
```

Error messages are defined in corresponding resource bundle, where each key matches the error name from enumeration.

The message can contain arbitrary number of parameters enclosed in `{}`.

invoiceplane_error.properties

```
LOGIN_FAILED=Login failed with user {0}
ADD_PRODUCT_FAILED=Add failed for product {0}
```

Now to make use of newly created exception simply throw `eu.ibagroup.easyrpa.engine.exception.EasyRpaException` and pass the arguments if the message requires them.

```
if (loginFailed) {
        throw new EasyRpaException(InvoicePlaneError.LOGIN_FAILED, credentials.getUser());
}
```

Built-in Exceptions

EasyRPA engine comes with a number of predefined exceptions.

They can be found grouped in three classes: `CoreError` , `CsError`, `NodeError` depending on scope and domain specifics.

All these classes are essentially enumerations implementing `eu.ibagroup.easyrpa.engine.exception.ErrorDetails` interface.

CoreError

These are exceptions related to inner workings of engine, such as issues with driver, data, AP execution etc.

Exception Code	Description
E1000	Unexpected fatal error
E1001	Unexpected error occurs
E1002	Booting error occurs
E1003	Booting error: module instantiation error
E1004	Unable to load a configuration file or no configuration file provided
E2000	Configuration key is not defined
E2001	Wrong configuration value
E2002	The specified configuration has wrong structure
E2003	Cannot find data
E2004	S3 is not configured
E2005	General configuration error
E3000	Execution error occurs in module
E3001	Execution error occurs in task
E3002	Instance creation error
E3003	Can not inject fields into instance
E3004	Can not execute annotated method for instance
E3005	Execution stopped at task
E5000	REST service call failed
E5001	Network timeout
E5002	REST call failed: standalone mode
E5003	An error occurs during uploading file to S3
E5004	An error occurs during generation of S3 url
E5005	An error occurs during downloading file from S3
E6000	Database consistency error
E6001	Database foreign constraint error
E6002	Database unique constraint error
E6003	Database duplicate record error
E6004	Datasource persistence error
E6005	Datasource SQL error
E6006	Datasource error: cannot determine table
E7000	Secret vault error

E7001	No trust store password found
E7002	No MongoDb password found
E7003	General security exception
E7004	Secret vault unsealing error
E7005	An error occurs during processing of file with secret configuration
E7006	Secret alias contains illegal characters
E8000	An error occurs during closing the driver
E8001	An error occurs during page creation
E8002	An error occurs during driver initialization
E8003	There is not current window for driver
E8004	Can't switch to window. Window with the following selector not found
E8005	An error occurs during opening application
E8006	An error occurs during taking screenshot
E8007	Timeout during waiting for UI element
E8008	Text validation error for UI element
E8009	List of opened windows is empty
E8010	Cannot find UI element on the screen
E8011	Can't launch application*/
E8012	Can't get windows
E8013	Unexpected driver error

CsError

This class comprises the exceptions derived from HTTP error codes and related to server-side issues such as access control, database integrity and others.

Exception Code	HTTP Code	Description
S1000	500	Unexpected error occurs
S4000	401	User is disabled
S4001	401	Invalid credentials for user
S4002	403	Access denied
S4003	404	Entity not found
S4004	400	Wrong input format
S4005	400	CSV error
S4006	400	Wrong input
S4007	400	Method arguments are not valid
S4008	400	Invalid cron expression
S4009	400	Wrong input format: type mismatch
S4010	400	Wrong input format: missing request parameter
S4011	401	User does not exist

S4012	401	Wrong token
S4013	400	Wrong input format: DataStore record
S4014	400	Wrong input: missing DataStore record in request
S4015	400	Wrong input: either schema or csv file is accepted but not both
S4016	400	Please provide valid JSON
S4017	400	Cannot extract zip package
S4018	400	Wrong structure of ap package
S4019	400	CSV header error
S3000	500	Asynchronous job exception
S6000	500	Database consistency error
S6001	500	Database foreign constraint error
S6002	500	Database unique constraint error
S6003	500	Database duplicate record error
S6004	500	Datasource persistence error
S6005	500	Datasource error
S6007	500	Automation process with the same name already exists
S6008	500	Data Store with the same name already exists
S6009	500	Schedule with the same name already exists
S6010	500	Node with the same name already exists
S6011	500	Role with the same name already exists
S6012	500	NodeConfiguration parameter with the same key already exists
S6013	500	Secret entry with the same alias already exists
S6014	500	User with the same username already exists
S6015	500	Capability with the same name already exists
S7000	500	Secret vault error

NodeError

These exceptions occur on node agent only

Exception Code	Description
N1000	Unexpected error
N2000	No configuration found
N3000	Multiply nodes
N4000	Communicate with another JVM failed

(v. 2.2) Lombok

Introduction

EasyRPA engine comes with built-in Lombok library. It is compile-time code generator which helps to get rid of repetitive code structures as well as unify the development practice across the team.

Any popular IDE today supports Lombok through plugins, read more about this in Install Lombok Plugin.

Lombok API includes stable and experimental features. It is recommended to stick to the stable since it is well tested, long supported and less likely to be changed.

ⓘ We only introduce the most popular Lombok applications, More of them can be found in the official documentation

Stable Features

@Slf4j

Generates slf4j logger field in a target class

```
@Slf4j
public class CalculatorMainPage extends ScreenPage {

    public void displayText() {
        log.info("Displaying text from calculator page");
    }
}
```

@Getter and @Setter

Generates getter and setter methods respectively for a target field

```
class Person {
    @Getter
        @Setter
    private String name;

        Person (String name) {
                this.name = name;
        }
}

class AppMain {
        public static void main(String[] args) {
                Person person = new Person("Ellie");
                System.out.println(person.getName());

                person.setName("Abby");
                System.out.println(person.getName());
        }
}
```

Annotations can also have modifications. One of them is `@Getter(lazy=true)` which allows lazy loading.

It means the underlying field will only be initalized when the getter is called first time and the result will be cached for later reuse.

@AllArgsConstructor

Generates the constructor receiving all the class members as arguments

```
@AllArgsConstructor
class Cat {
    private String name;
    private int age;
}

public static void main(String[] args) {
    Cat cat = new Cat("Felix", 8);
}
```

Lombok provides other annotations to generate constructors:

`@NoArgsConstructor` creates the default constructor i.e. without parameters

`@RequiredArgsConstructor` creates the constructor with only final and @NonNull fields

@ToString

As follows from its name, @ToString generates self-titled method.

```
class LocalRunner {
    @ToString
    static class Product {
        String name = "Butter";
        int price = 2;
    }

    public static void main(String[] args) {
        Product product = new Product();
        System.out.println(product.toString());
    }
}
```

If we run the above code, the output will look something as `LocalRunner.Product(name=Butter, price=2)` instead of `eu.ibagroup.easyrpa.lombok.LocalRunner$Product@121714c`

@Builder

The builder pattern simplifies the construction of class with large number of parameters especially when most of them are optional.

```
@Builder
public class NutritionFacts {
    private int servingSize = -1;
    private int servings = -1;
    private int calories = 0;
    private int fat = 0;
    private int sodium = 0;
    private int carbohydrate = 0;

    public static void main(String[] args) {
        NutritionFacts facts = NutritionFacts.builder()
                .calories(200)
                .fat(40)
                .carbohydrate(120)
                .build();
    }
}
```

As you see whith `@Builder` applied we don't have to declare all possible constructors of `NutritionFacts` covering different set of parameters. Instead we instantiate the class builder and only pass the needed parameters.

ⓘ There is much more of Lombok features you can read about in the offcial documentation.

Experimental Features

@UtilityClass

Utility class can't be instantiated and can only have static methods.

```
@UtilityClass
public class MathUtils {
    public int multiply(int a, int b) {
        return a * b;
    }
}
```

This is how this class looks after decompilation. The private constructor is added, the multiply method turned static and class itself turned final.

```
public final class MathUtils {
    private MathUtils() {
        throw new UnsupportedOperationException("This is a utility class and cannot be instantiated");
    }

    public static int multiply(int a, int b) {
        return a * b;
    }
}
```

We won't dive any deeper into experimental features since they are subject to change. but you can read more on them in the official documentation.

Delombok

Lombok annotations can be 'delomboked' i.e. unfolded to the actual code.

You can do it either by applying IDE plugin command or executing a command line.

To delombok in Intellij IDEA simply call `Refactor/Delombok/All lombok annotations` on a target class. Keep in mind that Lombok plugin must be installed.

(v. 2.2) Page Object Design

Page Object is a Design Pattern which has become popular in RPA for enhancing automation scripts maintenance and reducing code duplication. A page object is an object-oriented class that serves as an interface to a page of your application. The automation scripts then use the methods of this page object class whenever they need to interact with the UI of that page. The benefit is that if the UI changes for the page, the automation scripts themselves don’t need to change, only the code within the page object needs to change. Subsequently all changes to support that new UI are located in one place.

The Page Object Design Pattern provides the following advantages:

- There is a clean separation between automation scripts code and page specific code such as locators and layout.
- There is a single repository for the services or operations offered by the page rather than having these services scattered throughout the automation scripts.

Implementation

Page-Object approach supporting can achieved in the 3 following steps:

- Extend Application class
- Return a Page object from method "open"
- Extend appropriate Page class

Extend Application class

In the EasyRPA Page Object design we create an application class as entry point of any applications. All application classes should extend the **Application** class.

Our **Application** class has a constructor which takes a **Driver** object which will be responsible to initialization of pages via PageFactory (using the method "**createPage**"). Also as the entry point of any applications it has the method "**open**" which accept the any application arguments and should be responsible for opening application and returning the start page (usually it's login page).

So, the applications class which automation "Invoice Plane" web application will looks like:

InvoicePlaneApplication.java

```
import eu.ibagroup.easyrpa.engine.rpa.Application;
import eu.ibagroup.easyrpa.engine.rpa.driver.BrowserDriver;
import eu.ibagroup.easyrpa.engine.rpa.element.UiElement;

public class InvoicePlaneApplication extends Application<BrowserDriver, UiElement> {

    public InvoicePlaneApplication(BrowserDriver driver) {
        super(driver);
    }

    @Override
    public LoginPage open(String... args) {
                ...
    }
}
```

Return a Page object from method "open"

Method "open" should return a page object, so the whole **InvoicePlaneApplication.java** class will looks like the following:

InvoicePlaneApplication.java

```
import eu.ibagroup.easyrpa.engine.rpa.Application;
import eu.ibagroup.easyrpa.engine.rpa.driver.BrowserDriver;
import eu.ibagroup.easyrpa.engine.rpa.element.UiElement;
import eu.ibagroup.easyrpa.system.invoiceplane.page.LoginPage;

public class InvoicePlaneApplication extends Application<BrowserDriver, UiElement> {

    public InvoicePlaneApplication(BrowserDriver driver) {
        super(driver);
    }

    @Override
    public LoginPage open(String... args) {
        String invoicePlaneUrl = args[0];
        getDriver().get(invoicePlaneUrl);
        getDriver().manage().window().maximize();
        LoginPage loginPage = createPage(LoginPage.class);
        return loginPage;
    }
}
```

Method "**createPage**" is responsible for calling corresponding PageFactory to create a page, which is necessary to initialize the selector-annotations (such as @FindBy, @Wait). Also it saves the link to Application object which keeps the link to Driver object.

Extend appropriate Page class

Any class of page object should extend appropriate basic page:

- **WebPage** - if page works with **BrowserDriver**
- **DesktopPage** - if page works with **DesktopDriver**
- **ScreenPage** - if page works with **ScreenDriver**
- **SapPage** - if page works with **SapDriver**

In our example **LoginPage** should extends **WebPage** as it describe the page of "Invoice Plane" web application:

LoginPage

```
import eu.ibagroup.easyrpa.engine.model.SecretCredentials;
import eu.ibagroup.easyrpa.engine.rpa.page.WebPage;

public class LoginPage extends WebPage {

    public Dashboard login(SecretCredentials invoicePlaneCredentials) {
        ...
        return createPage(Dashboard.class);
    }

}
```

Pay attention that all new pages are created inside another pages should be initialized through "**createPage**" method, which is necessary to initialize the selector-annotations and to save the link to Application object which keeps the link to Driver object.

Selector-annotations

FindBy and Image annotations

All drivers have different limitation to use selector-annotations attributes. Use the compatibility matrix bellow to check which attributes can be used for the following annotations:

- `org.openqa.selenium.support.FindBy`
- `eu.ibagroup.easyrpa.engine.rpa.po.annotation.Image`

	@FindBy (id = "any")	@FindBy (name = "any")	@FindBy (className = "any")	@FindBy (css= "any")	@FindBy (tagName= "any")	@FindBy (linkText= "any")	@FindBy (partialLinkText = "any")	@FindBy (xpath = "any")	@Image (url= "any")
BrowserDriver									
DesktopDriver									
JavaDriver									
SapDriver									
ScreenDriver									

Wait annotations

Usually we use selector annotations (**FindBy** and **Image**) together with wait-annotations. There're 2 annotations to let driver know how should we wait for element:

- `eu.ibagroup.easyrpa.engine.rpa.po.annotation.Wait`
- `eu.ibagroup.easyrpa.engine.rpa.po.annotation.WithTimeout`

	@Wait	@WithTimeout
BrowserDriver		
DesktopDriver		
JavaDriver		
SapDriver		
ScreenDriver		

@Wait annotation

Annotation **@Wait** contains two attributes: to define a **wait timeout** (*default is 20 sec*) and to define a **wait function**.

So usage examples can be the following:

LoginPage

```
import eu.ibagroup.easyrpa.engine.rpa.page.WebPage;
import eu.ibagroup.easyrpa.engine.rpa.po.annotation.Wait;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;

public class MyPage extends WebPage {

    @FindBy(id = "email")
    @Wait(30)
    private WebElement emailField;

        ...
}
```

OR with a Wait function:

LoginPage

```
import eu.ibagroup.easyrpa.engine.rpa.page.WebPage;
import eu.ibagroup.easyrpa.engine.rpa.po.annotation.Wait;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;

public class MyPage extends WebPage {

    @FindBy(id = "email")
    @Wait(value = 30, waitFunc = Wait.WaitFunc.CLICKABLE)
    private WebElement emailField;

        ...
}
```

@WithTimeout annotation

Annotation **@WithTimeout** expects 2 attributes to be define: **time** (*required*) and **poll** (*default is 1 sec*)

Bellow is an example of usage:

LoginPage

```
import eu.ibagroup.easyrpa.engine.rpa.page.DesktopPage;
import eu.ibagroup.easyrpa.engine.rpa.po.annotation.WithTimeout;
import eu.ibagroup.easyrpa.engine.rpa.element.DesktopElement;
import org.openqa.selenium.support.FindBy;

public class MainPage extends DesktopPage {

    @FindBy(name = "password")
    @WithTimeout(time = 20, poll = 2)
    private DesktopElement passwordField;

        ...
}
```

Complete Page object example

Using the correct class structures and selector-annotations, complete **LoginPage.java** class for web application may looks like the following:

LoginPage

```
package eu.ibagroup.easyrpa.system.invoiceplane.page;

import eu.ibagroup.easyrpa.engine.annotation.AfterInit;
import eu.ibagroup.easyrpa.engine.exception.EasyRpaException;
import eu.ibagroup.easyrpa.engine.model.SecretCredentials;
import eu.ibagroup.easyrpa.engine.rpa.page.WebPage;
import eu.ibagroup.easyrpa.engine.rpa.po.annotation.Wait;
import eu.ibagroup.easyrpa.system.invoiceplane.exception.InvoicePlaneError;
import lombok.extern.slf4j.Slf4j;
import org.openqa.selenium.By;
import org.openqa.selenium.TimeoutException;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

@Slf4j
public class LoginPage extends WebPage {

        private static final String PAGE_CSS_SELECTOR = ".container #login";

        private static final long PAGE_LOAD_TIMEOUT = 20;

        private static final long LOGIN_WAIT_TIMEOUT = 20;

        private static final String successLoginCssSelector = "div#main-area";

        private static final String failedLoginCssSelector = "#login .alert";

        @FindBy(id = "email")
        @Wait(waitFunc = Wait.WaitFunc.CLICKABLE)
        private WebElement email;

        @FindBy(id = "password")
        @Wait(waitFunc = Wait.WaitFunc.CLICKABLE)
        private WebElement password;

        @FindBy(xpath = "//button[@type='submit']")
        @Wait(waitFunc = Wait.WaitFunc.CLICKABLE)
        private WebElement submit;

        @AfterInit
        public void init() {
                WebDriverWait wait = new WebDriverWait(getDriver(), PAGE_LOAD_TIMEOUT);
                wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector(PAGE_CSS_SELECTOR)));
        }

        public Dashboard login(SecretCredentials invoicePlaneCredentials) {
                email.click();
                email.clear();
                email.sendKeys(invoicePlaneCredentials.getUser());

                password.click();
                password.clear();
                password.sendKeys(invoicePlaneCredentials.getPassword());

                submit.click();

                try {
                        WebDriverWait wait = new WebDriverWait(getDriver(), LOGIN_WAIT_TIMEOUT);
                        wait.until(ExpectedConditions.or(ExpectedConditions.presenceOfElementLocated(By.
cssSelector(successLoginCssSelector)), ExpectedConditions.presenceOfElementLocated(By.cssSelector
(failedLoginCssSelector))));
                        boolean loginFailedAppeared = getDriver().findElements(By.cssSelector
(failedLoginCssSelector)).size() > 0;
                        if (loginFailedAppeared) {
                                throw new EasyRpaException(InvoicePlaneError.LOGIN_FAILED,
invoicePlaneCredentials.getUser());
                        }
                } catch (TimeoutException e) {
                        log.debug("Unknown error during InvoiceTO Place authorisation process.");
                        throw new RuntimeException();
                }

                return createPage(Dashboard.class);
        }
}
```

So **LoginPage** has only "**login**" method which accept the secret credentials. After it clicks button - there're 2 possible elements can be appeared: for success login flow and for failed login. So in the example above we create a wait condition as the combination of 2 possible conditions for different selectors. If login has done successfully - **LoginPage** returns **DashboardPage.**

Use your page-objects inside tasks (business-logic level)

After the page-objects structure is created, in your task you can easily start using it like the following:

```
InvoicePlaneApplication invoicePlaneApplication = new InvoicePlaneApplication(browserDriver);
LoginPage loginPage = invoicePlaneApplication.open(invoicePlaneUrl);
Dashboard dashboard = loginPage.login(invoicePlaneCredentials);
ProductsPage productsPage = dashboard.openProducts();
List<ProductTO> productsFromPage = productsPage.getProducts();
```

Additional rules

There is a lot of flexibility in how the page objects may be designed, but there are a few basic rules for getting the desired maintainability of your automation code.

- Page objects themselves should never contains any business specific logic. This is part of your automation scripts and should always be within the scripts's code, never in an page object. The page object will contain the representation of the page, and the services the page provides via methods but no code related to what is being automated should be within the page object.
- A page object does not necessarily need to represent all the parts of a page itself. The same principles used for page objects can be used to create "Page *Component* Objects" that represent discrete chunks of the page and can be included in page objects. These component objects can provide references the elements inside those discrete chunks, and methods to leverage the functionality provided by them. You can even nest component objects inside other component objects for more complex pages. If a page in the application you automate has multiple components, or common components used throughout the site (e.g. a navigation bar), then it may improve maintainability and reduce code duplication.

(v. 2.2) Drivers

Interface Driver

The interface Driver extends WebDriver class from the Selenium project, so inherits all standard Selenium driver methods. This interface is implemented by the following classes:

- BrowserDriver - to work with web-applications
- DesktopDriver - to work with desktop applications (except Java application)
- JavaDriver - to work with Java desktop applications
- SapDriver - to work with SAP applications
- ScreenDriver - to work with the any applications on image-based approach

Some of the methods from Driver interface are supported in all of above drivers, but some of them may not be supported in some driver.

In addition of standard Selenium WebDriver methods this Driver has the following useful methods:

- initPage
- sleep
- getScreenshot
- getScreenshotAsBytes
- getScreenshotAsFile
- getScreenshotAsBase64
- clipboard
- waitForElement
- waitForElementNot
- waitFor
- getInputDevices

Opening new applications

get(String)

BrowserDriver - native Selenium support

DesktopDriver - supports (please refer to driver page to see additional details)

JavaDriver - supports (please refer to driver page to see additional details)

SapDriver - method is not supported

ScreenDriver - method is not supported

The first thing you will want to do after launching a bot is to open your application. This methods is used to open an application / website and it accepts the executable file path / website link (depends on driver type).

```
public void get(String url)
```

Browser navigation

getTittle

BrowserDriver - native Selenium support

DesktopDriver - supports

JavaDriver - supports

SapDriver - method is not supported

ScreenDriver - method is not supported

You can read the current application title (or browser title).

```
public String getTitle()
```

getCurrentUrl

BrowserDriver - native Selenium support

DesktopDriver - method is not supported

JavaDriver - please refer to driver page to see details

SapDriver - method is not supported

ScreenDriver - method is not supported

You can read the current URL from the browser’s address bar using this method.

```
public String getCurrentUrl()
```

navigate

BrowserDriver - native Selenium support

DesktopDriver - method is not supported

JavaDriver - method is not supported

SapDriver - method is not supported

ScreenDriver - method is not supported

This method returns an abstraction (Navigation object) which contains the helpful methods to perform navigation in Browser.

```
public Navigation navigate()
```

```
driver.navigate().to("https://selenium.dev"); //longer way of driver.get("https://selenium.dev")
driver.navigate().back();
driver.navigate().forward();
driver.navigate().refresh();
```

Working with windows

switch to window

⚠ Because of drivers nature, each driver has its own way to switch to windows:

BrowserDriver - please refer to driver page to see details

DesktopDriver - please refer to driver page to see details

JavaDriver - please refer to driver page to see details

SapDriver - please refer to driver page to see details

ScreenDriver - is not supported

getWindowHandle

⚠ **BrowserDriver** - supports

DesktopDriver - supports

JavaDriver - supports

SapDriver - method is not supported

ScreenDriver - method is not supported

Each window has a unique identifier which remains persistent in a single session. You can get the window handle of the current window by using this method.

```
public String getWindowHandle()
```

getWindowHandles

⚠ **BrowserDriver** - supports

DesktopDriver - supports

JavaDriver - supports

SapDriver - method is not supported

ScreenDriver - method is not supported

Return a set of window handles which can be used to iterate over all open windows available for driver so you can use handle to switch to window by passing them to "switch to window" methods.

```
public Set<String> getWindowHandles()
```

close

BrowserDriver - supports

DesktopDriver - supports

JavaDriver - supports

SapDriver - supports (please refer to driver page to see additional details)

ScreenDriver - method is not supported

Close the current window (quitting the browser if it's the last window currently open for BrowserDriver).

```
public void close()
```

window management

BrowserDriver - supports

DesktopDriver - supports

JavaDriver - supports

SapDriver - supports

ScreenDriver - method is not supported

You can get the Window object to manage it (to change and get the size, change position). It returns the interface for managing the current window

```
public Window window()
```

Each driver has its own Window interface implementation. Please, refer driver specific page for more details:

- (v. 2.2) Browser Driver Window
- (v. 2.2) Desktop Driver Window
- (v. 2.2) Java Driver Window
- (v. 2.2) Sap Driver Window

Take screenshots

BrowserDriver - supports

DesktopDriver - supports

JavaDriver - supports

SapDriver - supports

ScreenDriver - supports

All drivers support getting a screenshot of the whole screen. Captured screenshot can be returned in different format.

getScreenshotAsBytes

```
public byte[] getScreenshotAsBytes()
```

getScreenshotAsFile

```
public File getScreenshotAsFile()
```

getScreenshotAsBase64

```
public String getScreenshotAsBase64()
```

Also the same methods can be used from each particular element to capture screenshot of an element. Please, refer the driver specific page for more details:

- (v. 2.2) Browser Driver Element
- (v. 2.2) Desktop Driver Element
- (v. 2.2) Java Driver Element
- (v. 2.2) Sap Driver Element
- (v. 2.2) Screen Driver Element

Working with UI Elements

Each driver has its own specific type of elements and specific way to build the search query (By object). So it's needed to see additional details from driver specific pages.

Bellow are details which locators are supported by which driver (**eu.ibagroup.easyrpa.engine.rpa.locator.By**)

Selector type	BrowserDriver	DesktopDriver	JavaDriver	SapDriver	ScreenDriver
By.id					
By.name					
By.xpath					
By.ccsSelector					
By.className					
By.tagName					
By.linkText					
By.partialLinkText					
By.desktopSearch					
By.image					
By.anchor					
By.sapId					

BrowserDriver - native Selenium support (please refer to driver page to see additional details)

DesktopDriver - supports (please refer to DesktopDriver page to see additional details)

JavaDriver - supports (please refer to driver page to see additional details)

SapDriver - supports (please refer to driver page to see additional details)

ScreenDriver - supports (please refer to driver page to see additional details)

findElement

It is used to find an element and returns a first matching single WebElement reference, that can be used for future element actions.

```
public UiElement findElement(By by)
```

findElements

Similar to "findElement", but returns a list of matching UI Elements. To use a particular UI Element from the list, you need to loop over the list of elements to perform action on selected element.

Each driver has its own specific type of elements and specific way to build the search query (By object). So it's needed to see additional details from driver specific pages.

```
public List<UiElement> findElement(By by)
```

waitForElement

Similar to "findElement", but it doesn't throw an exception immediately in case of not element was found. It will wait for specified timeout until element appears. Also supports optional "suppress exception" parameter which suppress an exception in case of "true" value and method returns "null" instead.

Also instead of search query ("By" object) method may accept Function object. It's used for conditional queries (e.g. *ExpectedCondition s.visibilityOfElementLocated(By.xpath("xpath_string")*)

```
public UiElement waitForElement(By by, int secondsToPoll)
public UiElement waitForElement(By by, int secondsToPoll, boolean suppressTimeoutException)

public UiElement waitForElement(Function<WebDriver, UiElement> function, int secondsToPoll)
public UiElement waitForElement(Function<WebDriver, UiElement> function, int secondsToPoll, boolean
suppressTimeoutException)
```

Keyboard, Mouse, Clipboard

BrowserDriver - supports

DesktopDriver - supports

JavaDriver - supports

SapDriver - supports

ScreenDriver - supports

All drivers supports input devices such as Keyboard and Mouse. Also there is an access to system clipboard from all drivers.

getInputDevices

Method which is supported by all drivers returns an abstraction which lets you access to Mouse and Keyboard devices (has appropriate getters).

```
public InputDevices getInputDevices()
```

For example:

```
import eu.ibagroup.easyrpa.engine.rpa.interactions.InputDevices;
import eu.ibagroup.easyrpa.engine.rpa.interactions.Mouse;
import eu.ibagroup.easyrpa.engine.rpa.interactions.Keyboard;

InputDevices inputDevices = driver.getInputDevices();
Mouse mouse = inputDevices.getMouse();
Keyboard keyboard = inputDevices.getKeyboard();
```

getKeyboard

This method returns an abstraction (Keyboard object) which allows you to perform an actions with the keyboard, such as sending, pressing and releasing keys.

```
public Keyboard getKeyboard()
```

The short example of how to press the "Ctrl + S" combination which is usually responsible for saving:

```
import eu.ibagroup.easyrpa.engine.rpa.interactions.Keyboard;
import org.sikuli.hotkey.Keys;

Keyboard keyboard = driver.getInputDevices().getKeyboard();
keyboard.pressKey(Keys.CTRL);
keyboard.sendKeys("s");
keyboard.releaseKey(Keys.CTRL);
```

Keyboard class uses Sikuli library to send a keys, so it's important to use the keys constants from **org.sikuli.hotkey.Keys**.

Also it's important to have the same **keyboard language selected in Windows** tray menu as the **characters language** you send to keyboard methods. Otherwise you'll get an exception

getMouse

This method returns an abstraction (Mouse object) which allows you to perform an actions with the mouse, such as right and left buttons clicking, mouse moving, doubleclicking.

```
public Mouse getMouse()
```

Mouse object has various type of methods to perform clicking, mouse moving and etc. which accept arguments such as "x" and "y" integer coordinates, Point object. Also mouse moving can be perform with offset:

```
public void click(Point where)
public void click(int x, int y)

public void doubleClick(Point where)
public void doubleClick(int x, int y)

public void contextClick(Point where)
public void contextClick(int x, int y)

public void mouseDown(Point where)
public void mouseDown(int x, int y)

public void mouseUp(Point where)
public void mouseUp(int x, int y)
public void mouseMove(Point where)
public void mouseMove(int x, int y)
public void mouseMove(Point where, long xOffset, long yOffset)
public void mouseMove(int x, int y, long xOffset, long yOffset)
```

Bellow is an example of how to perform mouse click and moving with offset:

```
driver.getInputDevices().getMouse().click(10, 10);
driver.getInputDevices().getMouse().mouseMove(10, 10, 40, 40);
```

clipboard

This method returns an abstraction (Cliboard object) which allows you to get, set and clean clipboard text.

```
public Clipboard clipboard()
```

Clipboard object contains the following methods:

```
public String getText()
public void setText(String text)
public void clean()
```

The short example of how get value from clipboard:

```
String valueFromClipboard = driver.clipboard().getText();
```

(v. 2.2) Browser Driver

This page describes only Browser Driver specific features and methods which may be different in other drivers.

Implementation which is common for all drivers is described in Drivers page. Please, read it before.

BrowserDriver is used to automate web browser actions.

This driver is based on the Selenium WebDriver which drives a browser natively, as a user would. In addition to Selenium WebDriver, EasyRPA BrowserDriver brings additional useful methods.

Driver Initialization

You can initialize the driver in your automation process by the following way:

```
@Driver(value = DriverParams.Type.BROWSER, param = {
        @DriverParameter(key = DriverParams.Browser.SELENIUM_NODE_CAPABILITIES, initializer = BrowserDriver.
DefaultChromeOptions.class) })
private BrowserDriver browserDriver;
```

Please notice that we pass `DefaultChromeOptions` class as capability initializer in the above example, so we expect that selenium grid will provide the chrome node for us. If we want to use the browser other than Chrome, we need to pass the corresponding capability initializer instead.

All the supported browsers are listed in the table below

Target Browser	Capability Initializer
Chrome	BrowserDriver.DefaultChromeOptions.class
Edge	BrowserDriver.DefaultEdgeOptions.class
Firefox	BrowserDriver.DefaultFirefoxOptions.class
Internet Explorer	BrowserDriver.DefaultIeOptions.class
Opera	BrowserDriver.DefaultOperaOptions.class

Driver Params

Param name	Measure	Default Value	Description
DriverParams. Browser.**SELENIUM_HUB_URL**	URL	`http://localhost:4444/wd/hub`	Selenium Hub URL.
DriverParams. Browser.**SELENIUM_NODE_CAP**	`org. openqa.`	null	Used to set properties of browsers to perform browser automation of web applications. It stores the capabilities as key-value pairs and these capabilities are used to set browser properties like browser name, browser version, path of browser driver in the system, etc. to determine the behavior of browser

ABILITIES	`selenium.Capabilities`		at run time. E.g. you can read about Chrome browser capabilities by the following link.
DriverParams.Browser.**PAGE_LOAD_TIMEOUT_SECONDS**	seconds	1800	Limits the time that the script allots for a web page to be displayed. If the page loads within the time then the script continues. If the page does not load within the timeout the script will be stopped by a *TimeoutException*.
DriverParams.Browser.**IMPLICITLY_WAIT_TIMEOUT_SECONDS**	seconds	0	This timeout is used to specify the amount of time the driver should wait while searching for an element if it is not immediately present.

Bellow is an example of how to put Chrome options capabilities:

```
ChromeOptions chromeOptions = new ChromeOptions();

HashMap<String, Object> chromePrefs = new HashMap<>();
chromePrefs.put("profile.default_content_settings.popups", 0);
chromePrefs.put("download.default_directory", filePath);

chromeOptions.setExperimentalOption("prefs", chromePrefs);

browserDriver = getDriver(BrowserDriver.class, new HashMap<Enum, Object>() {{
    put(DriverParams.Browser.SELENIUM_NODE_CAPABILITIES, chromeOptions);
}});
```

Opening new applications

The first thing you will want to do after launching a browser is to open your website. This can be achieved in a single line:

```
browserDriver.get("https://selenium.dev");
```

Window management

Get window handle

WebDriver does not make the distinction between windows and tabs. If your site opens a new tab or window, Selenium will let you work with it using a window handle. Each window has a unique identifier which remains persistent in a single session. You can get the window handle of the current window by using:

```
driver.getWindowHandle();
```

Switching windows or tabs

Browser driver supports switching to window by **name** or **window handle**. For that it has the following method allowing you to switch to window:

```
public void switchToWindow(String nameOrHandle);
```

Also it supports methods which help you to wait until window appears and switch to it only after that. It returns "true" if it switched successfully:

```
public boolean waitAndSwitchToWindow(String windowSearch, int secondsToPoll);
public boolean waitAndSwitchToWindow(String windowSearch, By by, int secondsToPoll);
public boolean waitAndSwitchToWindow(String windowSearch, By by, int secondsToPoll, boolean
suppressTimeoutException)
```

Working with UI Elements

Inspector

UI Elements represents a DOM elements. Browser Driver API provides built-in methods to find the elements which are based on different properties like ID, Name, Class, XPath, CSS Selectors, link Text, etc. To construct a selector you can use built-in browser DOM inspector.

E.g. you can read about Chrome browser inspector by the following link.

Find elements

Please, check the table which locators are supported by BrowserDriver:

Selector type	BrowserDriver
By.id	
By.name	
By.xpath	
By.ccsSelector	
By.className	
By.tagName	
By.linkText	
By.partialLinkText	
By.desktopSearch	
By.image	
By.anchor	
By.sapId	

Bellow an example of how to find element by name:

```
WebElement webElement = browserDriver.findElement(By.name("q"));
```

You can read more information of how to work with selenium selectors:

XPath Tutorial

CSS selectors

Locators Tutorial

Selenium documentation

(v. 2.2) Browser Driver Element

WebElement represents a DOM element. WebElements can be found by searching from the document root using a WebDriver instance, or by searching under another WebElement. You can read more documentation about WebElement on Selenium project documentation.

Get Active Element

It is used to track (or) find DOM element which has the focus in the current browsing context.

```
// Get attribute of current active element
String attr = driver.switchTo().activeElement().getAttribute("title");
```

Is Element enabled

This method is used to check if the connected Element is enabled or disabled on a webpage. Returns a boolean value, **True** if the connected element is **enabled** in the current browsing context else returns **false**.

```
  //returns true if element is enabled else returns false
  boolean value = driver.findElement(By.name("btnK")).isEnabled();
```

Is Element Selected

This method determines if the referenced Element is *Selected* or not. This method is widely used on Check boxes, radio buttons, input elements, and option elements.

Returns a boolean value, **True** if referenced element is **selected** in the current browsing context else returns **false**.

```
//returns true if element is checked else returns false
boolean value = driver.findElement(By.cssSelector("input[type='checkbox']:first-of-type")).isSelected();
```

Get Element Tag Name

It is used to fetch the TagName of the referenced Element which has the focus in the current browsing context.

```
//returns TagName of the element
String value = driver.findElement(By.cssSelector("h1")).getTagName();
```

Get Element Rect

It is used to fetch the dimensions and coordinates of the referenced element.

The fetched data body contain the following details:

- X-axis position from the top-lef corner of the element
- y-axis position from the top-lef corner of the element

- Height of the element
- Width of the element

```
// Returns height, width, x and y coordinates referenced element
Rectangle res = driver.findElement(By.cssSelector("h1")).getRect();

// Rectangle class provides getX,getY, getWidth, getHeight methods
System.out.println(res.getX());
```

Get Element CSS Value

Retrieves the value of specified computed style property of an element in the current browsing context.

```
// Retrieves the computed style property 'color' of linktext
String cssValue = driver.findElement(By.linkText("More information...")).getCssValue("color");
```

Get Element Text

Retrieves the rendered text of the specified element.

```
  // Retrieves the text of the element
  String text = driver.findElement(By.cssSelector("h1")).getText();
```

(v. 2.2) Browser Driver Window

Screen resolution can impact how your web application renders, so WebDriver provides mechanisms for moving and resizing the browser window.

Current window object can be retrieved from driver using the method:

```
Window window = driver.manage().window();
```

Get window size

Fetches the size of the browser window in pixels.

```
//Access each dimension individually
int width = driver.manage().window().getSize().getWidth();
int height = driver.manage().window().getSize().getHeight();

//Or store the dimensions and query them later
Dimension size = driver.manage().window().getSize();
int width1 = size.getWidth();
int height1 = size.getHeight();
```

Set window size

Restores the window and sets the window size.

```
driver.manage().window().setSize(new Dimension(1024, 768));
```

Get window position

Fetches the coordinates of the top left coordinate of the browser window.

```
// Access each dimension individually
int x = driver.manage().window().getPosition().getX();
int y = driver.manage().window().getPosition().getY();

// Or store the dimensions and query them later
Point position = driver.manage().window().getPosition();
int x1 = position.getX();
int y1 = position.getY();
```

Set window position

Moves the window to the chosen position.

```
// Move the window to the top left of the primary monitor
driver.manage().window().setPosition(new Point(0, 0));
```

Maximize window

Enlarges the window. For most operating systems, the window will fill the screen, without blocking the operating system's own menus and toolbars.

```
driver.manage().window().maximize();
```

Minimize window

Minimizes the window of current browsing context. The exact behavior of this command is specific to individual window managers.

Minimize Window typically hides the window in the system tray.

```
driver.manage().window().minimize();
```

Fullscreen window

Fills the entire screen, similar to pressing F11 in most browsers.

```
driver.manage().window().fullscreen();
```

(v. 2.2) Desktop Driver

> This page describes only Desktop Driver specific features and methods which may be different in other drivers.
>
> Implementation which is common for all drivers is described in Drivers page. Please, read it before.

DesktopDriver is used to automate desktop applications.

This driver is based on the Mmarquee UIAutomation project which a Java-based wrapper Microsoft UIAutomation Library. This framework is for automating rich client applications based on Win32 (including Delphi), WPF and other Windows applications (including Java SWT). It uses the JNA library to make calls to the COM-based WIndows automation library.

Driver Initialization

You can initialize the driver in your automation process by the following way:

```
@Driver(DriverParams.Type.DESKTOP)
private DesktopDriver desktopDriver;
```

Driver Params

Param name	Measure	Default Value	Description
DriverParams.Desktop.**LAUNCH_APPLICATION_TIMEOUT**	ms	5000	Parameter is responsible for timeout which driver should wait after it called the command to start application until it's started.

```
@Driver(value = DriverParams.Type.DESKTOP, param = {
        @DriverParameter(key = DriverParams.Desktop.LAUNCH_APPLICATION_TIMEOUT, direct = "50000")
})
private DesktopDriver desktopDriver;
```

Opening new applications

Desktop Driver supports 2 methods to start new application: using path or path with arguments.

```
public void get(String path)
public void get(String... command)
```

Bellow is an example of how to start application using path:

```
driver.get("C:\\Windows\\system32\\calc.exe");
```

```
//or

driver.get("notepad.exe");
```

Window management

Desktop Driver supports switching to window by various combination using window **class name**, **title** or **title regexp**, **window handle**. For that it has the following methods allowing you to switch to window:

```
public void switchToWindow(String titleOrHandle)
public void switchToWindow(Pattern titleRegexp)
public void switchToWindow(String className, String titleOrHandle)
public void switchToWindow(String className, Pattern titlePattern)
```

Also it supports methods which help you to wait until window appears and switch to it only after that, with the various combination of input parameters. It returns "true" if it switched successfully:

```
/**
 * Switches to the desktop 'window' associated with the class name and title regexp (or window handle).
 *
 * @param className Class name to search for.
 * @param titleRegexp Regexp for title to search for or window handle.
 * @param titleOrHandle title to search for or window handle.
 * @param secondsToPoll Seconds to wait for windows to be appeared.
 * @param suppressTimeoutException Set true to don't let method throw an exception, so it returns "false" value
instead.
 *
 * @return True if driver switched successfully
 *
*/
public boolean waitAndSwitchToWindow(String titleOrHandle, int secondsToPoll)
public boolean waitAndSwitchToWindow(Pattern titleRegexp, int secondsToPoll)
public boolean waitAndSwitchToWindow(String titleOrHandle, int secondsToPoll, boolean suppressTimeoutException)
public boolean waitAndSwitchToWindow(Pattern titleRegexp, int secondsToPoll, boolean suppressTimeoutException)
public boolean waitAndSwitchToWindow(String className, String titleOrHandle, int secondsToPoll)
public boolean waitAndSwitchToWindow(String className, Pattern titleRegexp, int secondsToPoll)
public boolean waitAndSwitchToWindow(String className, String titleOrHandle, int secondsToPoll, boolean
suppressTimeoutException)
public boolean waitAndSwitchToWindow(String className, Pattern titlePatternOrHandle, int secondsToPoll, boolean
suppressTimeoutException)
```

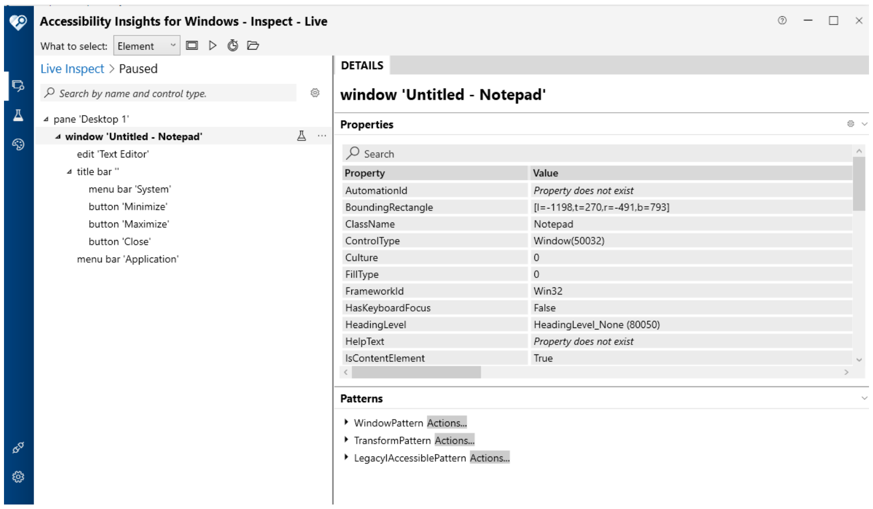
Example:

```
driver.switchToWindow("Untitled - Notepad");
```

Working with UI Elements

Inspector

To automate applications using DesktopDriver, we use the Accessibility Insights for Windows tool. This tool allows to get information about interface elements such as Name, ID, Type, etc.


Accessibility Insights for Windows - Inspect - Live
What to select: Element
Live Inspect > Paused
Search by name and control type.
DETAILS

pane 'Desktop 1'
window 'Untitled - Notepad'
edit 'Text Editor'
title bar ''
menu bar 'System'
button 'Minimize'
button 'Maximize'
button 'Close'
menu bar 'Application'

window 'Untitled - Notepad'

Properties

Property	Value
AutomationId	*Property does not exist*
BoundingRectangle	[l=-1198,t=270,r=-491,b=793]
ClassName	Notepad
ControlType	Window(50032)
Culture	0
FillType	0
FrameworkId	Win32
HasKeyboardFocus	False
HeadingLevel	HeadingLevel_None (80050)
HelpText	*Property does not exist*
IsContentElement	True

Patterns

- WindowPattern Actions...
- TransformPattern Actions...
- LegacyIAccessiblePattern Actions...

To download Accessibility Insights for Windows, or to learn more about it, click **here**.

Find elements

Before you switch to any windows using driver, the desktop (root) window is considered as the active one. As the desktop window is the root window for all elements and for all other windows on the screen - searching will be perform slower if you don't switch to any specific window.

So when you call "findElement(s)" method it's strongly **recommended to switch to target window** before, to don't have the root window as the active one.

Please, check the table which locators are supported by DesktopDriver:

Selector type	DesktopDriver
By.id	
By.name	
By.xpath	
By.ccsSelector	
By.className	
By.tagName	
By.linkText	
By.partialLinkText	
By.desktopSearch	
By.image	
By.anchor	
By.sapId	

Desktop driver supports only "**By.desktopSearch**" element locator. This locator accept special **UIQuery** object which you can build providing necessary element attributes.

Bellow an example of how to find element by name, class name and control type (all information can be extracted by Inspector):

```
DesktopElement desktopElement = driver.findElement(By.desktopSearch(UIQuery.builder()
        .name("Text Editor")
        .className("Edit")
        .controlType(ControlType.Edit.getValue())
        .build()
));
```

(v. 2.2) Desktop Driver Element

DesktopElement represents a UI element on the screen. DesktopElement can be found by searching using a DesktopDriver instance. **Ge tting DesktopElement by searching under another DesktopElement is not supported yet**.

DesktopDriver API provides built-in methods to find the DesktopElement which are based on different properties like ID, Name, Text, etc.

Is Element enabled

This method is used to check if the connected Element is enabled or disabled on a page. Returns a boolean value, **True** if the connected element is **enabled** in the current browsing context else returns **false**.

```
//returns true if element is enabled else returns false
boolean value = driver.findElement(By.desktopSearch(UIQuery.builder().className("Edit")).isEnabled();
```

Is Element Selected

This method determines if the referenced Element is *Selected* or not. This method is widely used on Check boxes, radio buttons, input elements, and option elements.

Returns a boolean value, **True** if referenced element is **selected** in the current browsing context else returns **false**.

```
//returns true if element is checked else returns false
boolean value = driver.findElement(By.desktopSearch(UIQuery.builder().className("Checkbox")).isSelected();
```

Get Element Tag Name (Control Type)

It is used to fetch the Control Type value of the referenced Element. "Tag name" in context of Desktop Driver means the **ControlType** attribute value.

```
//returns ControlType of the element
String value = driver.findElement(By.desktopSearch(UIQuery.builder().name("Text Editor")).getTagName();
```

Get Element Rect

It is used to fetch the dimensions and coordinates of the referenced element.

The fetched data body contain the following details:

- X-axis position from the top-lef corner of the element
- y-axis position from the top-lef corner of the element
- Height of the element
- Width of the element

```
// Returns height, width, x and y coordinates referenced element
Rectangle res =  driver.findElement(By.desktopSearch(UIQuery.builder().name("Text Editor")).getRect();
// Rectangle class provides getX,getY, getWidth, getHeight methods
System.out.println(res.getX());
```

Get Element Text

Retrieves the rendered text of the specified element.

```
// Retrieves the text of the element
String text = driver.findElement(By.desktopSearch(UIQuery.builder().className("Edit")).getText();
```

(v. 2.2) Desktop Driver Window

- Maximize
- Focus
- Get Size
- Get Position

Current window object can be retrieved from driver using the method:

```
Window window = driver.manage().window();
```

DesktopWindow supports only the following methods:

Maximize

Maximize the current window.

```
driver.manage().window().maximize();
```

Focus

Calling of this method is focusing on current window.

```
((DesktopWindow) driver.manage().window()).focus();
```

Get Size

Returns the Dimension object with the "width" and "height" properties

```
Dimension windowDimension = driver.manage().window().getSize();

System.out.println("Current window size is: " + windowDimension.getWidth() + "x" + windowDimension.getHeight());
```

Get Position

Returns the current position of the window on screen. The value is returned as "Point" object with "x" and "y" properties.

```
Point windowPosition = driver.manage().window().getPosition();

System.out.println("Current window position is: " + windowPosition.getX() + "x" + windowPosition.getY());
```

(v. 2.2) Java Driver

This page describes only Java Driver specific features and methods which may be different in other drivers.

Implementation which is common for all drivers is described in Drivers page. Please, read it before.

Java driver automates applications with java UI, that is built from of `java.awt.*` package. This for example includes Swing and Oracle Forms

Java Driver consists of EasyRPA **driver** implementation and **java agent**

Java agent is a jar package implementing JsonWire protocol. It attaches to target java application and starts http server

Java Driver implementation is a part of engine. It makes use of JsonWire protocol while providing a familiar selenium-like programming interface to developer

Prepare node for Java Automation

The following manual steps should be done to prepare node for Java automation

1. if you are running Automation process manually, specify the path to java agent jar - **easy-rpa-java-agent.jar** in system properties. Node agent does it automatically.

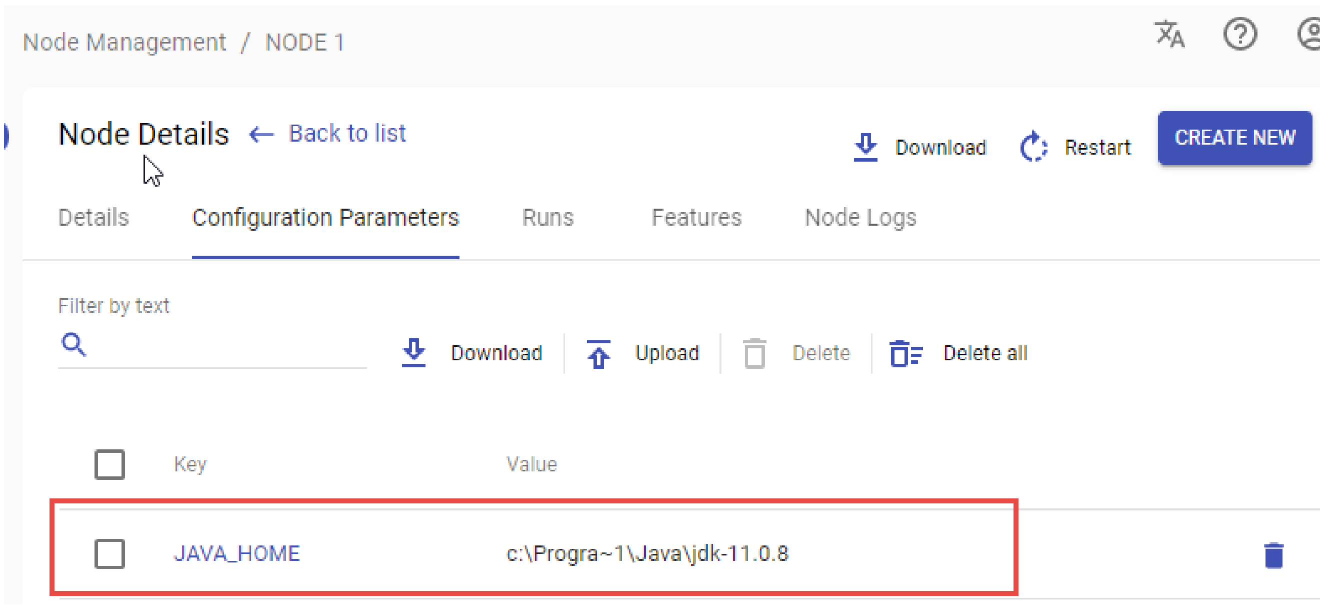
```
// is equivalent to VM option -Dmarathon.agent.file="D:/path/to/easy-rpa-java-agent.jar"
System.setProperty("marathon.agent.file", "D:/path/to/easy-rpa-java-agent.jar");
```

2. if you are running Automation process manually, specify the path to java agent jar - **easy-rpa-java-agent.jar** in system properties. Node agent does it automatically.

java.policy must be updated by the policy grant for the easy-rpa-java-agent.jar. The jar path must be the same as you specified in p.1. For node agents runs, you need refer to jar that is located in node agent package.

```
grant codeBase "file:/C:/<Path to node directory>/easy-rpa-java-agent.jar" {
   permission java.security.AllPermission;
};
```

3. Java runtime for Automation Process run and Java application you are automating must be the same. By default node agent uses the same java runtime for AP run as specified for start the itself. To change the java runtime for automation process, you need to define java home parameter in node configuration:



Key	Value
JAVA_HOME	c:\Progra~1\Java\jdk-11.0.8

CPU architecture must match: if target application runs under java x64 then AP must be also started under java x64

Driver Initialization

You can initialize the driver in your automation process by the following way:

```
@Driver(DriverParams.Type.JAVA)
private JavaDriver jDriver;
```

Opening new applications

We have a specific article about opening applications in Java Driver so please refer to (v. 2.2) How to launch or connect to Java Application

Browser navigation

getCurrentUrl

returns window (i.e. java top UI container) properties as JSON string

```
// returns {"title":"InputVerificationDemo","tagName":"window","component.class.name":"javax.swing.JFrame"}
String url = driver.getCurrentUrl();
```

Read more about navigation in JavaDriver Basic Navigation

Window management

getTitle

calls `getTitle()` on focused `java.awt.Frame` or `java.awt.Dialog`

```
String title = driver.getTitle();
```

getWindowHandle

returns identity hashcode of `java.awt.Window` (java top UI container) encoded as hex string

```
// returns "7abde323"
String handle = driver.getWindowHandle();
```

getWindowHandles

returns all window handles registered in `sun.awt.AppContext`

```
Set<String> handles = driver.getWindowHandle();
```

calls `dispose()` on current `java.awt.Window`

```
drive.close();
```

switch to window

Java driver can only switch windows inside a single Java application. To take over another application the one must instantiate one more driver

Java Driver supports switching to window by title or handle

- switches to window (java top UI container) by its title

```
driver.switchTo().window("Login Success");
```

- successively switches to each window by its handle

```
Set<String> windowHandles = getDriver().getWindowHandles();
for (String handle : windowHandles ) {
        driver.switchTo().window(handle);
}
```

Working with UI Elements

Inspector

To inspect Java applications we suggest using Java inspector tool which is described in (v. 2.2) How to use Java Inspector

Find elements

The element locators supported by Java Driver:

Selector type	DesktopDriver
By.id	
By.name	
By.xpath	

By.ccsSelector	
By.className	
By.tagName	
By.linkText	
By.partialLinkText	
By.desktopSearch	
By.image	
By.anchor	
By.sapId	

`By.id` and `By.name` search for value from `java.awt.Component.getName`

```
// these are the same
driver.findElement(By.id("username"));
driver.findElement(By.name("username"));
```

`By.className` searches by fully qualified java class name including all subclasses

```
// returns the elements of class javax.swing.JPasswordField
driver.findElements(By.className("javax.swing.JPasswordField"));

// returns the elements of class javax.swing.JTextField including JPasswordField
// because JPasswordField extends JTextField
driver.findElements(By.className("javax.swing.JTextField"));
```

EasyRPA implements CSS finder for Java applications. You can use JavaDriver's findElement(s) call to quickly find a component in the application

Selector	Description	Example
tagname	Selects an element with the given tagname. A tagname is computed by finding the Swing/AWT superclass of the component and converting CamelCase to camel-case	`driver.findElement(By.tagName ("text-field")); // JTextField driver.findElement(By.tagName ("spinner")); // JSpinner`
*	Gets all the elements, retuns list of elements	`driver.findElements(By. cssSelector("*"));`
.	Returns the same element	`// tree and tree_1 are same tree = driver.findElements(By. cssSelector("tree")); tree_1 = tree.find_element(By. cssSelector("."));`
#name	Select a component with the given name	

		`// the element has name "loanAmount" set through java.awt.Component#setName driver.findElement(By.cssSelector("#loanAmount"));`
[attribute =value]	Find components with the given value for the attribute. The operation can be *= (contains), /= (regex match), = (equals), ^= (startswith), $= (endswith).	`// finds all the buttons whose text is equal to "Click Me" driver.findElement(By.cssSelector("button[text='Click Me']"));`
:	Find components for which returns true. Pseudoclasses avaiable are `selected`, `enabled`, `displayed`, `hidden` and `instance-of("")`	`// finds all the text-fields which are enabled driver.findElement(By.cssSelector("text-field:enabled"));`
::	find the pseudo elements	`// returns list of all options from JComboBox driver.findElement(By.cssSelector("combo-box::all-options"));`

More components and selectors that can be used for it both with examples on ruby can be found here: https://marathontesting.com/marathonite-user-guide/java-swing-components/

(v. 2.2) How to launch or connect to Java Application

Introduction

Currently Java driver supports a number of launch modes for target application

Each of them has its own specifics so let's take a closer look

Starting new java application

Launch from JAR

When launching target application from jar, `LaunchMode.EXECUTABLE_JAR` must be passed as first argument and the path to application jar as second

```
jDriver.get(JavaDriver.LaunchMode.EXECUTABLE_JAR, "D:\\path\\to\\swing-application.jar");
```

Launch from JNLP

JNLP is deprecated since Java 9 and removed since 11. We recommend to run JNLP under Java 8.

Initial setup must be perfomed in order to automate JNLP applications

- to escape warnings like "Your Java version is out of date":

Open the file `%userprofile%\AppData\LocalLow\Sun\Java\Deployment\deployment.properties` and add a line in the beggining:

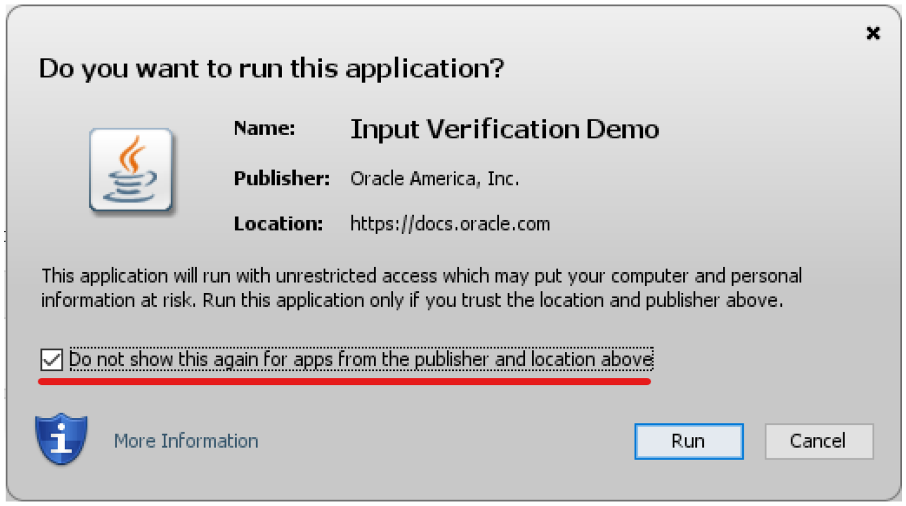
deployment.properties

```
deployment.expiration.check.enabled=false
```

- if self-signed application is blocked from running by security settings:

In Windows, open Control Panel->Java->Security->Edit Site List... and add the path to jnlp file, e.g. `file:///D:/InputVerificationDemo.jnlp`

On the first run the security prompt will appear, check "Do not show this again..." and then run


Do you want to run this application?

Name: Input Verification Demo

Publisher: Oracle America, Inc.

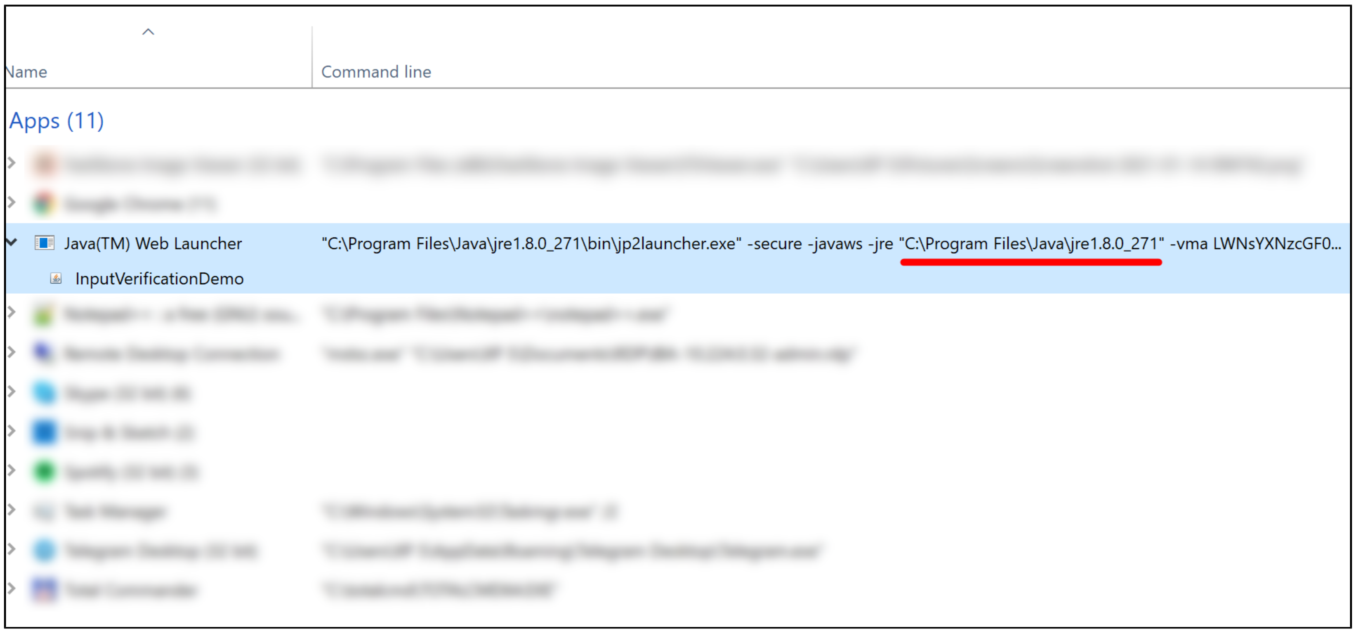
Location: https://docs.oracle.com

This application will run with unrestricted access which may put your computer and personal information at risk. Run this application only if you trust the location and publisher above.

☑ Do not show this again for apps from the publisher and location above

- if java agent attaches to target application but fails to start JsonWire server:

You have to check `javaws.policy` file. To find it first determine the path to JRE which runs your jnlp applications.
Run jnlp file manually then check Windows Task Manager. Locate your jnlp application under Java Web Launcher process, `-jre` argument from command line will tell you the path you need.



Name	Command line
Apps (11)	
Java(TM) Web Launcher	"C:\Program Files\Java\jre1.8.0_271\bin\jp2launcher.exe" -secure -javaws -jre "C:\Program Files\Java\jre1.8.0_271" -vma LWNsYXNzcGF0...
InputVerificationDemo	

Now go to JRE folder - according to the above screen it is `C:\Program Files\Java\jre1.8.0_271`
`javaws.policy` is in `lib\security\` folder. So the ultimate path is `c:\Program Files\Java\jre1.8.0_271\lib\security\javaws.policy`

Open `javaws.policy` in text editor and add the snippet to the end of file:

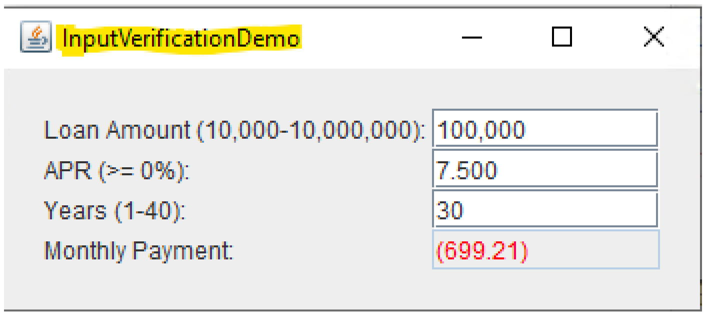
java.policy

```
grant codeBase "file:/path/to/easy-rpa-java-agent.jar" {
    permission java.security.AllPermission;
  };
```

When all is set and done, the jnlp application can be started by passing arguments to `driver.get` method:

- `LaunchMode.JAVA_WEBSTAR`
- path to jnlp file
- start window title

The start window title is basically the title on top of the window which normally comes first after java loading screen dissapears



Loan Amount (10,000-10,000,000): 100,000
APR (>= 0%): 7.500
Years (1-40): 30
Monthly Payment: (699.21)

The final code can look like following:

```
jDriver.get(JavaDriver.LaunchMode.JAVA_WEBSTART, "D:\\InputVerificationDemo.jnlp", "InputVerificationDemo");
```

Launch from Command Line

Java driver is also capable to start java application from command line. To do so, simply pass `LaunchMode.COMMAND_LINE` and path to command line script to `driver.get` method

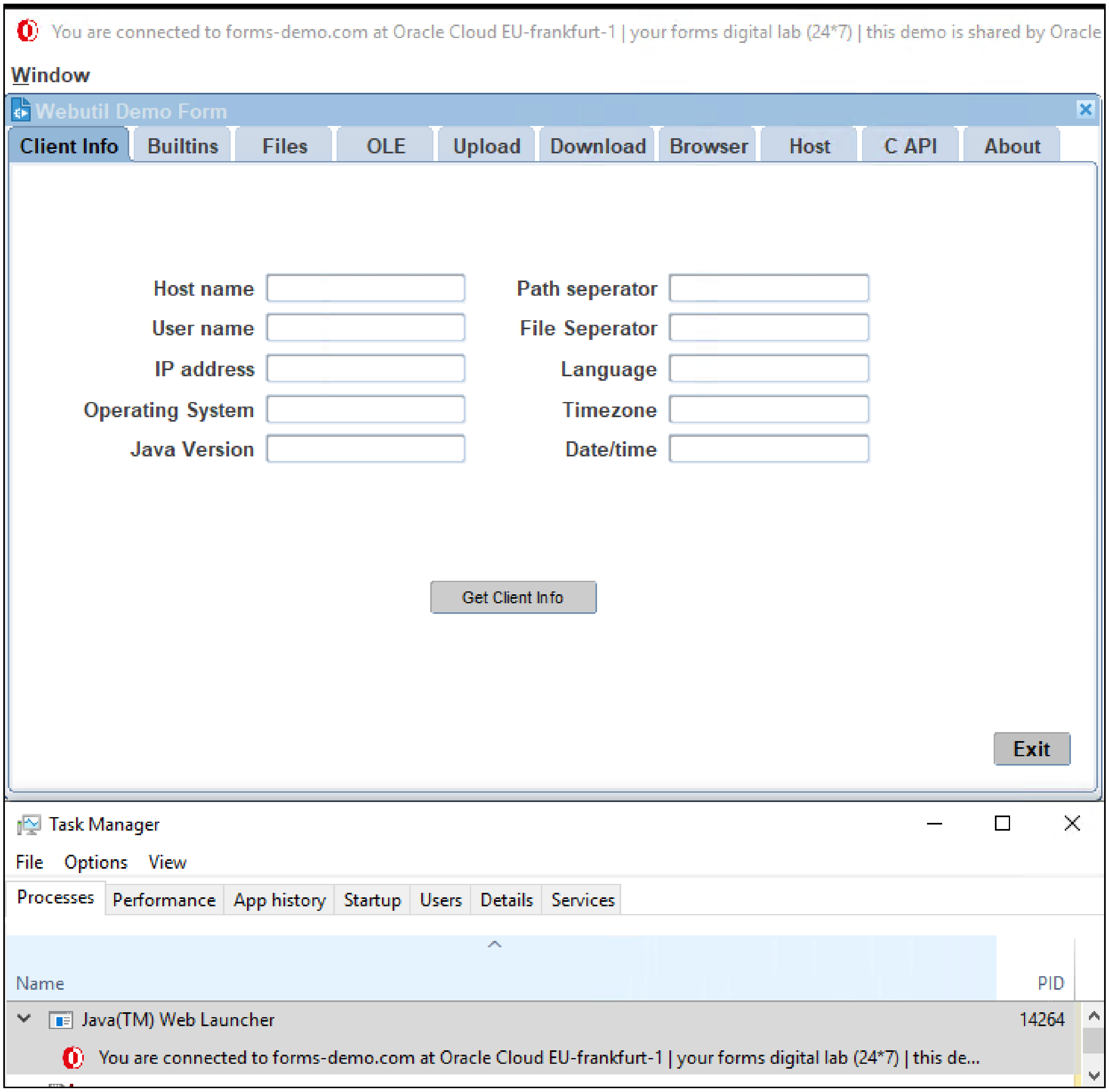
```
jDriver.get(JavaDriver.LaunchMode.COMMAND_LINE, "D:\\run-application.bat");
```

Attach to running java application

Java driver offers capability to attach to already running java application using it's name when starting a new application instance simultaneously with driver is not the case

To figure out the application name start it manually and find its process id using your tool of choice

For example here we started webutl demo from Oracle and found its PID is 14264 in Windows task manager


You are connected to forms-demo.com at Oracle Cloud EU-frankfurt-1 | your forms digital lab (24*7) | this demo is shared by Oracle

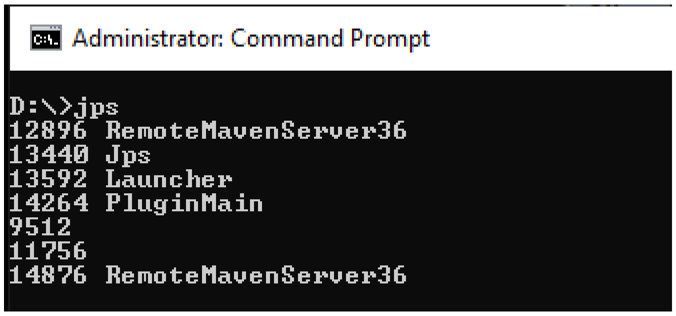
Window

Webutil Demo Form

Host name		Path seperator	
User name		File Seperator	
IP address		Language	
Operating System		Timezone	
Java Version		Date/time	

Name	PID
Java(TM) Web Launcher	14264
You are connected to forms-demo.com at Oracle Cloud EU-frankfurt-1 | your forms digital lab (24*7) | this de...	

Next we need to start jps utility from jdk in command line to find the application name corresponding to 14264


Administrator: Command Prompt

```
D:\>jps
12896 RemoteMavenServer36
13440 Jps
13592 Launcher
14264 PluginMain
9512
11756
14876 RemoteMavenServer36
```

Here we go, the application name is `PluginMain`. Let's pass it to `driver.get` method together with `LaunchMode.JAVA_ATTACH`

```
jDriver.get(JavaDriver.LaunchMode.JAVA_ATTACH, "PluginMain");
```

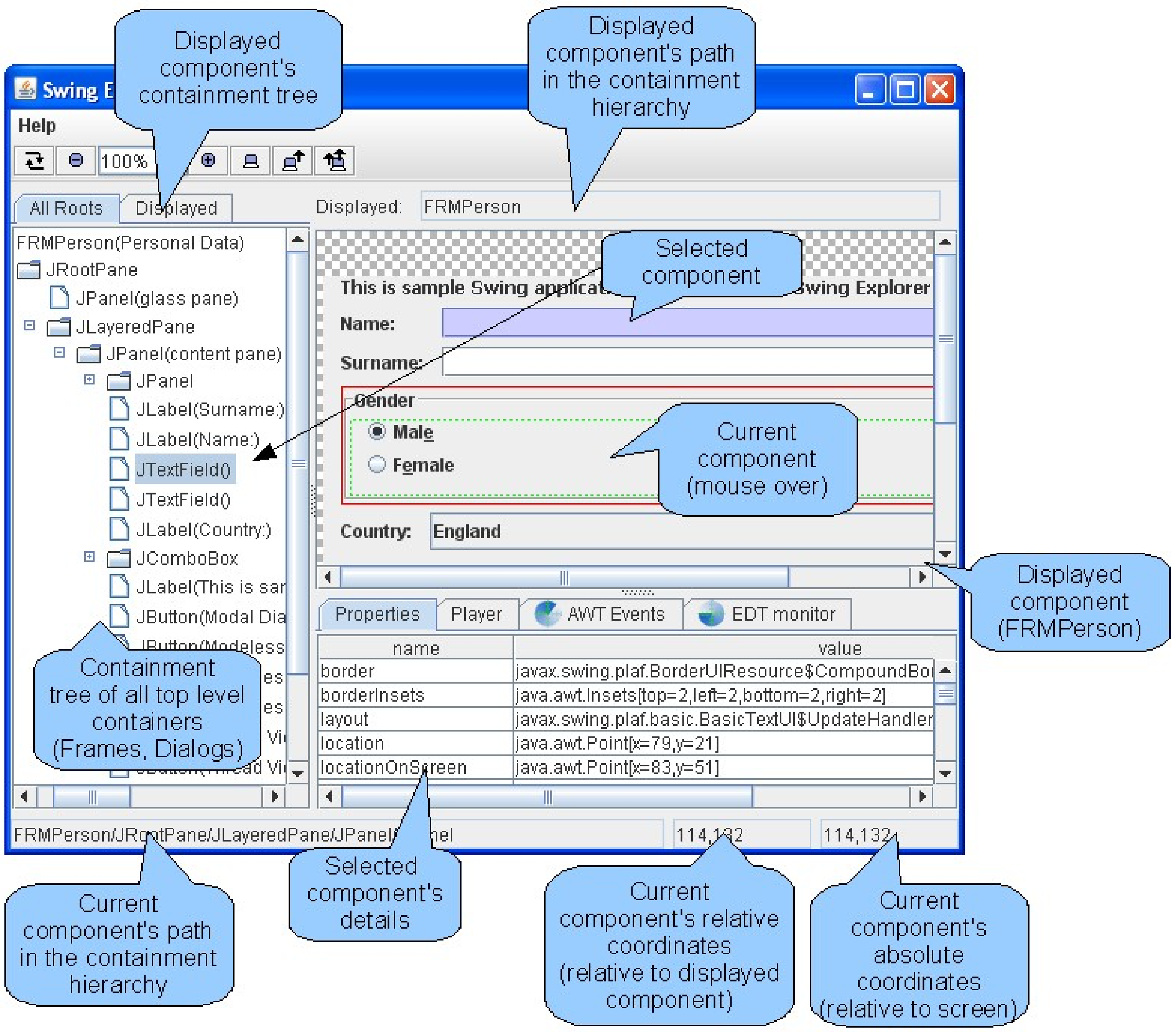
(v. 2.2) How to use Java Inspector

Introduction

Java Inspector is a tool for inspecting Java Swing GUIs. It works in a similar way as developer plugins for HTML browsers but for Java Swing toolkit.

With Swing Explorer you can visually browse through application component hierarchy.

Below is shown how basic Swing application is inspected by Java Inspector.



Displayed component's containment tree
Displayed component's path in the containment hierarchy
Selected component
Current component (mouse over)
Displayed component (FRMPerson)
Containment tree of all top level containers (Frames, Dialogs)
Current component's path in the containment hierarchy
Selected component's details
Current component's relative coordinates (relative to displayed component)
Current component's absolute coordinates (relative to screen)
Swing E
Help
100%
All Roots
Displayed
Displayed: FRMPerson
FRMPerson(Personal Data)
JRootPane
JPanel(glass pane)
JLayeredPane
JPanel(content pane)
JPanel
JLabel(Surname:)
JLabel(Name:)
JTextField()
JTextField()
JLabel(Country:)
JComboBox
JLabel(This is sar
JButton(Modal Dia
This is sample Swing applicat
Swing Explorer
Name:
Surname:
Gender
Male
Female
Country:
England
Properties
Player
AWT Events
EDT monitor
FRMPerson/JRootPane/JLayeredPane/JPanel/
114,132
114,132

name	value
border	javax.swing.plaf.BorderUIResource$CompoundBor
borderInsets	java.awt.Insets[top=2,left=2,bottom=2,right=2]
layout	javax.swing.plaf.basic.BasicTextUI$UpdateHandler
location	java.awt.Point[x=79,y=21]
locationOnScreen	java.awt.Point[x=83,y=51]

Download Java Inspector

Please find inspector here: easy-rpa-java-inspector

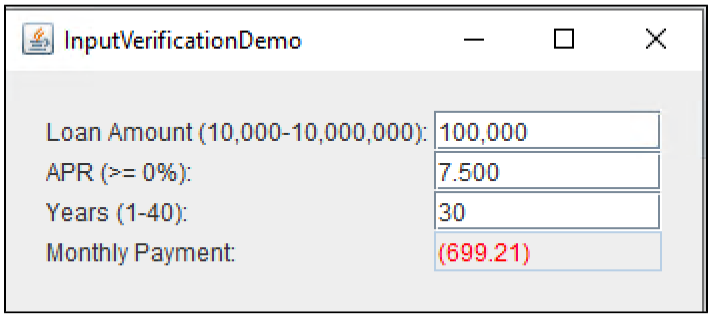
Starting Java Inspector

To launch the inspector use the `startup` command, found in root directory.

Inspector is started under default version of java which is subject to your operating system - it can be x64, x86 or other. If you need to inspect the application running under a certain OS architecture then please modify java path in `startup.bat` file

When started, the first screen of Java Inspector displays the list of Java proccesses. Not all of them can be inspected so it is user responsibility to correctly define a required application.

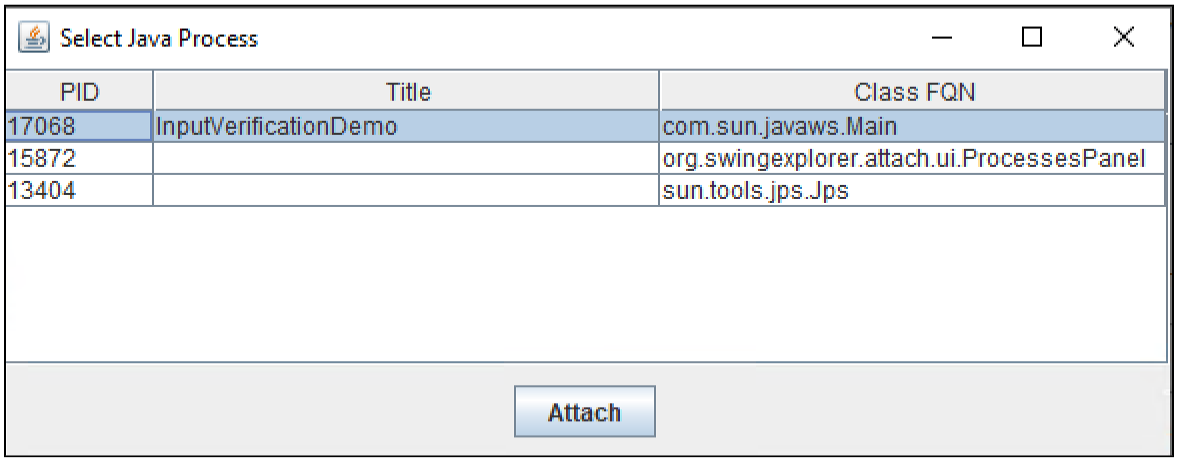
Consider an example: let's start InputVerification found in Oracle documentation



InputVerificationDemo

Loan Amount (10,000-10,000,000):	100,000
APR (>= 0%):	7.500
Years (1-40):	30
Monthly Payment:	(699.21)

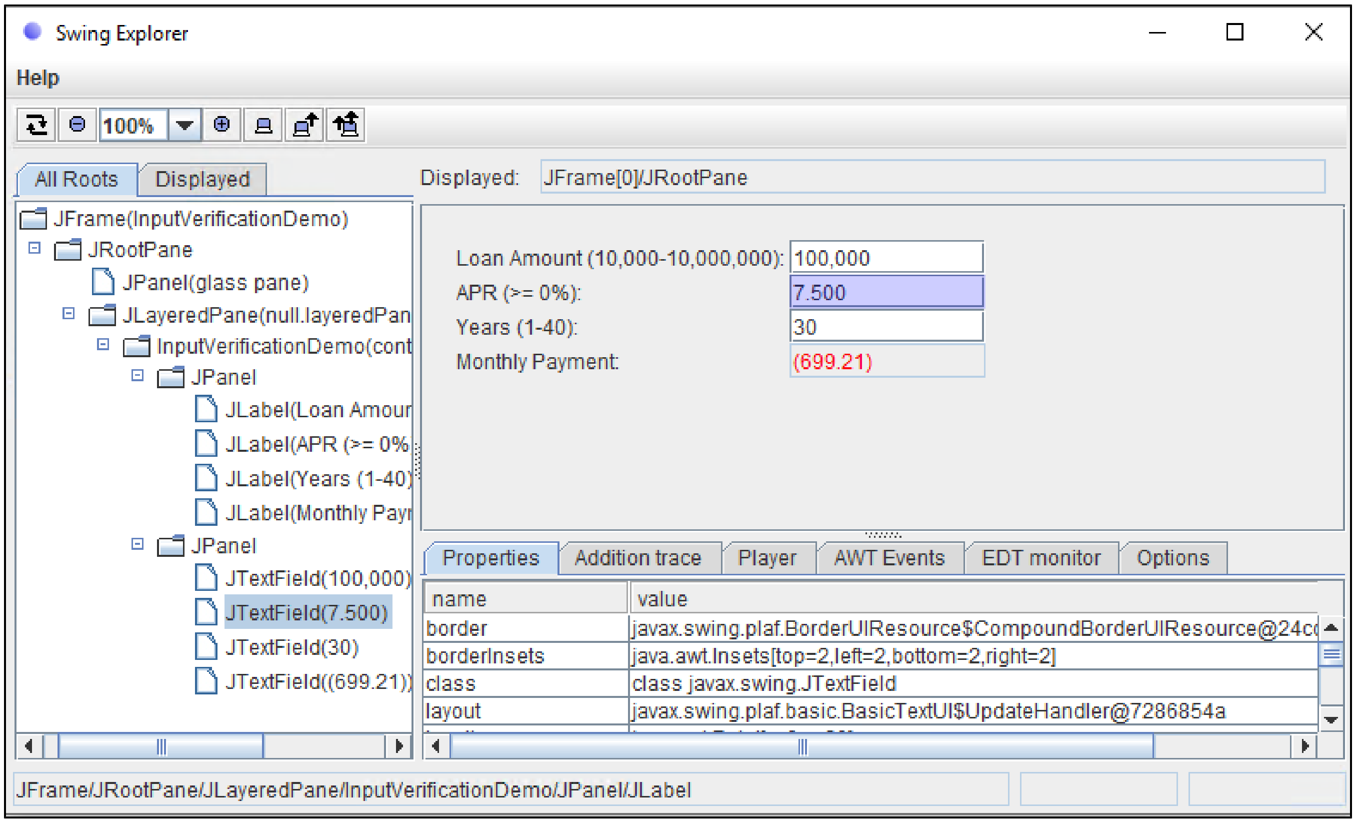
Now start Java Inspector using `startup`



Select Java Process

PID	Title	Class FQN
17068	InputVerificationDemo	com.sun.javaws.Main
15872		org.swingexplorer.attach.ui.ProcessesPanel
13404		sun.tools.jps.Jps

Even though InputVerification is a single Java desktop application running on the machine, we still see other java proccesses. Clearly the one we are in is the first list with title "InputVerificationDemo". Let's go and attach to it



name	value
border	javax.swing.plaf.BorderUIResource$CompoundBorderUIResource@24c
borderInsets	java.awt.Insets[top=2,left=2,bottom=2,right=2]
class	class javax.swing.JTextField
layout	javax.swing.plaf.basic.BasicTextUI$UpdateHandler@7286854a

After inspector is attached, the explorer window appears with tree of elements rendered in the left pane

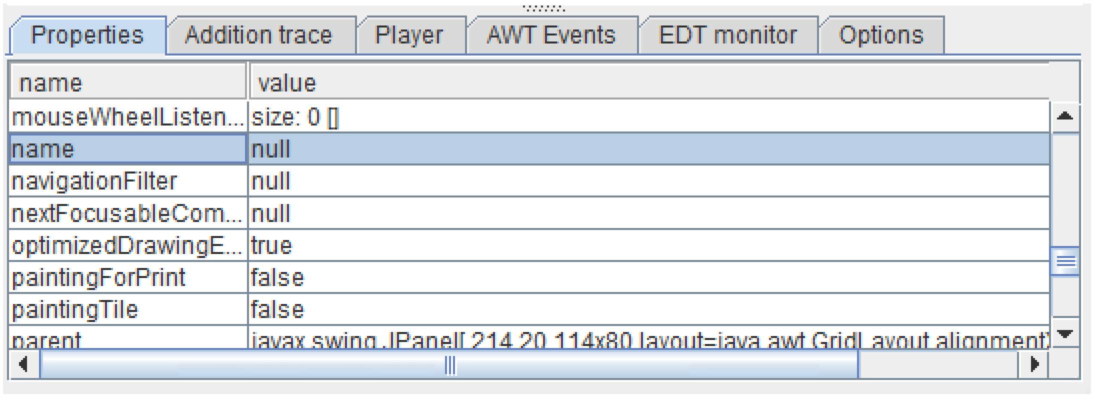
Double click on tree element will display the element inside inspector and single click will highlight it

You can vice-versa select it straight in the display and the element will be automatically highlighted in the tree

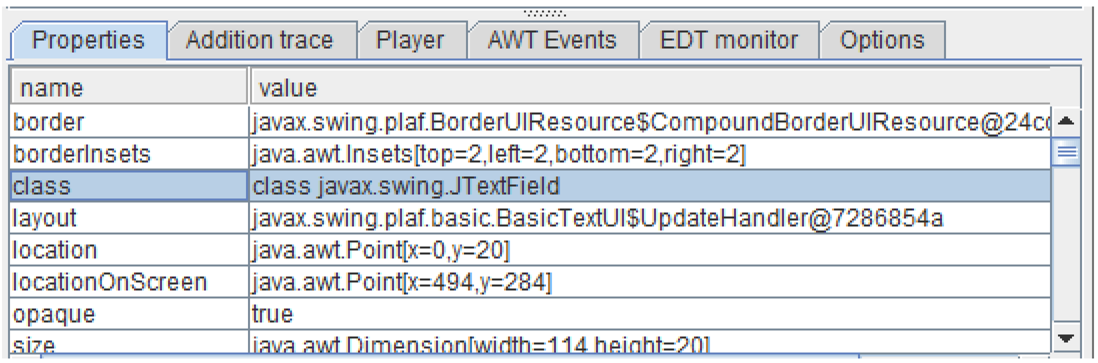
Retreiving Element Selector

Say we want our robot to change the value of APR, this is the field we selected on last screen.

It's typical to use element `name` or `class` to find element, so let's check the Properties panel to see its values



name	value
mouseWheelListen...	size: 0 []
name	null
navigationFilter	null
nextFocusableCom...	null
optimizedDrawingE...	true
paintingForPrint	false
paintingTile	false
parent	javax.swing.JPanel[,214,20,114x80,layout=java.awt.GridLayout,alignment



name	value
border	javax.swing.plaf.BorderUIResource$CompoundBorderUIResource@24c
borderInsets	java.awt.Insets[top=2,left=2,bottom=2,right=2]
class	class javax.swing.JTextField
layout	javax.swing.plaf.basic.BasicTextUI$UpdateHandler@7286854a
location	java.awt.Point[x=0,y=20]
locationOnScreen	java.awt.Point[x=494,y=284]
opaque	true
size	java.awt.Dimension[width=114,height=20]

Since we don't have unique name for this element and there are multiple elements on current screen with the same class name, we'll find all the elements and retreive the one we needed by index

```
        List<WebElement> elements = driver.findElements(By.className("javax.swing.JTextField"));

        WebElement loanAmountInput = elements.get(0);
        WebElement aprInput = elements.get(1); // this our APR field
        WebElement yearsInput = elements.get(2);
        WebElement paymentInput = elements.get(3);

                // now let's change its value
        aprInput.clear();
        aprInput.sendKeys("12.5");
```

(v. 2.2) Java Driver Element

What is Java Driver element?

It is a good old WebElement from Selenium except that it communicates to java agent instead of vendor driver such as chromedriver.
exe

Most of its methods act similar to Selenium WebElement so we only take a look at the differences

Instance Methods

submit

calls `stopEditing()` on `javax.swing.JTree` which saves any edits that are currently in progress on a cell

getTagName

returns hyphenated class name of element

```
WebElement element = driver.findElement(By.className("javax.swing.JTextField"))
String tagName = element.getTagName(); //returns "text-field"
```

Not implemented

These methods are not implemented and will throw `org.openqa.selenium.UnsupportedCommandException`:

```
getRect
getCssValue
isSelected
getScreenshotAs
```

(v. 2.2) Java Driver Window

Get window size

Calls `getSize()` on `javax.swing.JFrame`

```
java.awt.Dimension size = driver.manage().window().getSize();
```

Set window size

Calls `setSize` on `javax.swing.JFrame`

```
driver.manage().window().setSize(new Dimension(600, 300));
```

Get window position

Calls `getLocation()` on `javax.swing.JFrame`

```
org.openqa.selenium.Point = driver.manage().window().getPosition();
```

Set window position

Calls `setLocation` on `javax.swing.JFrame`

```
driver.manage().window().setPosition(new Point(10, 10));
```

Maximize window

Calls `setExtendedState(Frame.MAXIMIZED_BOTH)` on `javax.swing.JFrame` if the one is supported by window

```
driver.manage().window().maximize();
```

(v. 2.2) SAP Driver

This page describes only Sap Driver specific features and methods which may be different in other drivers.

Implementation which is common for all drivers is described in Drivers page. Please, read it before.

Prepare node for SAP Automation

SAP driver automates the actions are performed against SAP desktop client. Under the hood SAP driver employs ActiveX Component which requires manual steps need to be done ones by node administrator to prepare node for SAP automation:

- Setup SAP GUI For Windows 7.50 according to the https://www.sap.com/documents/2017/05/bc8d025a-b97c-0010-82c7-eda71af511fa.html
- Apply registry patch **sap_driver_fix.reg** from node agent package
- Install Microsoft Visual C++ Redistributable 2015-2019 x64 and/or x86

You also need to setup SAP Client connections that could be done by SAP user.

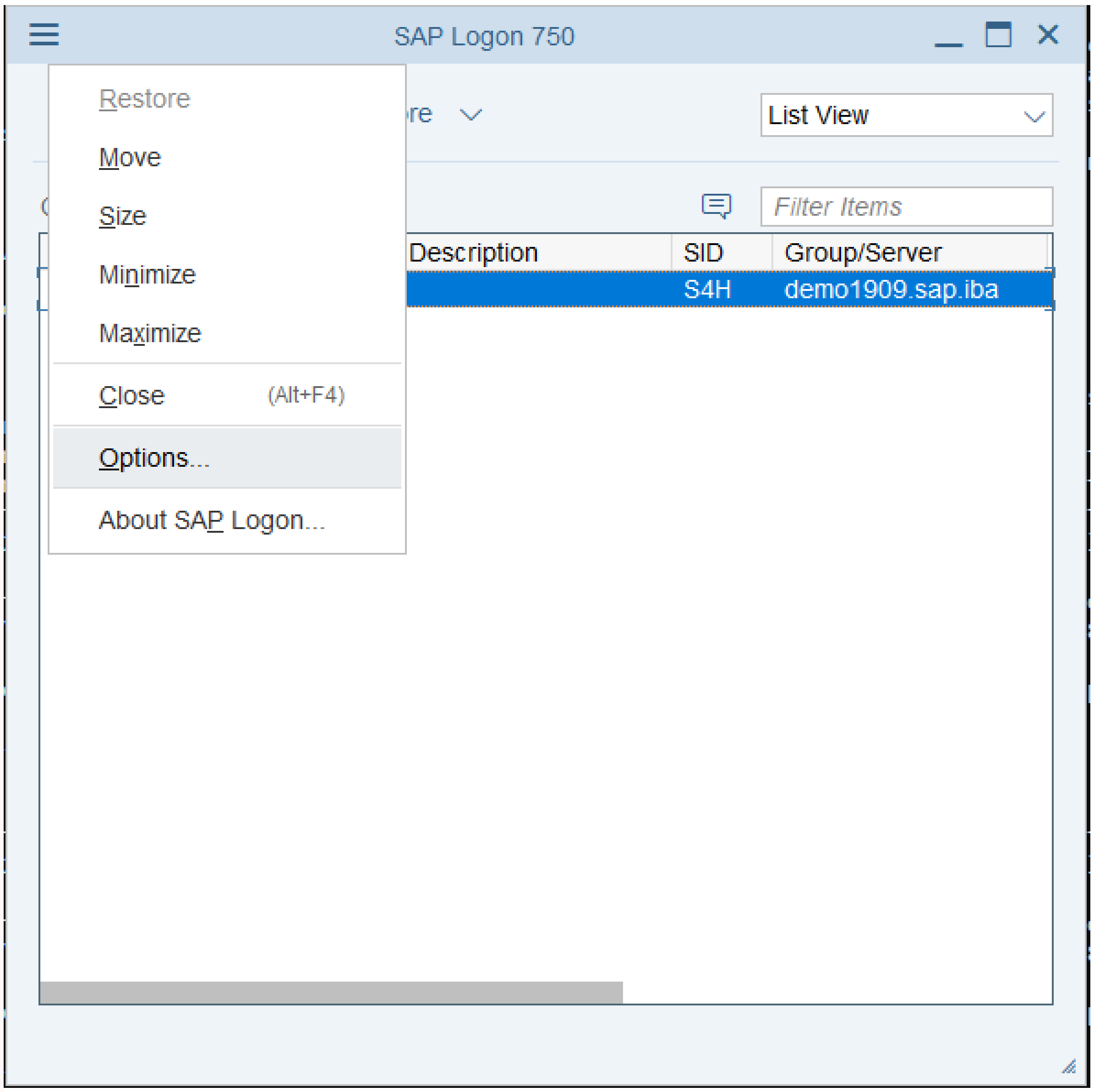
Prepare your SAP Server for RPA

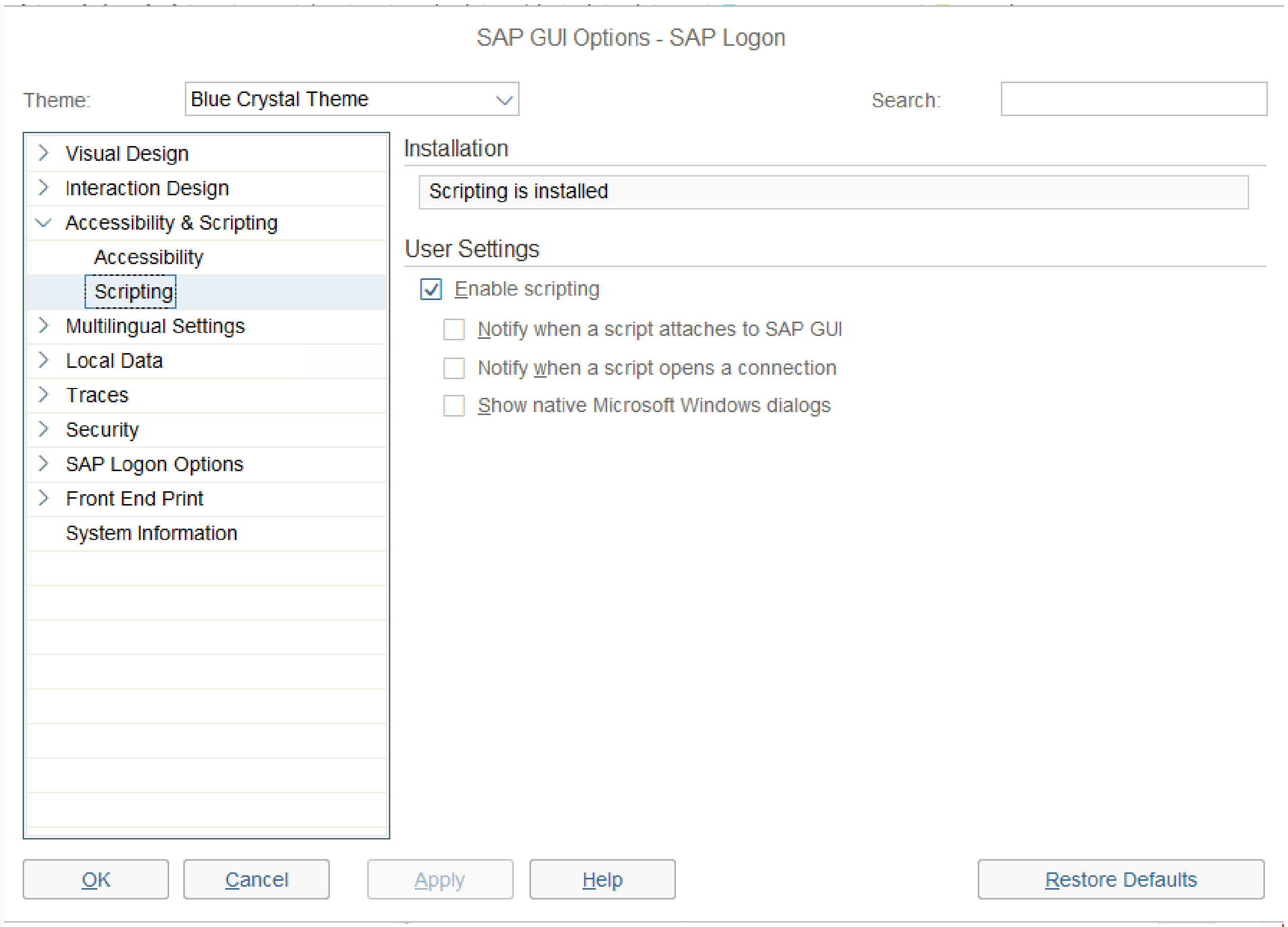
Enable SAP Scripting on SAP Server

It is recommended to download Tracker from https://tracker.stschnell.de/

One must be able to run Tracker against SAP Frontend and see its object tree after completing this step

Enable scripting on client side, switch off the notifications:


SAP Logon 750
Restore
Move
Size
Minimize
Maximize
Close (Alt+F4)
Options...
About SAP Logon...
List View
Filter Items
Description
SID
Group/Server
S4H
demo1909.sap.iba



Theme: Blue Crystal Theme

Installation

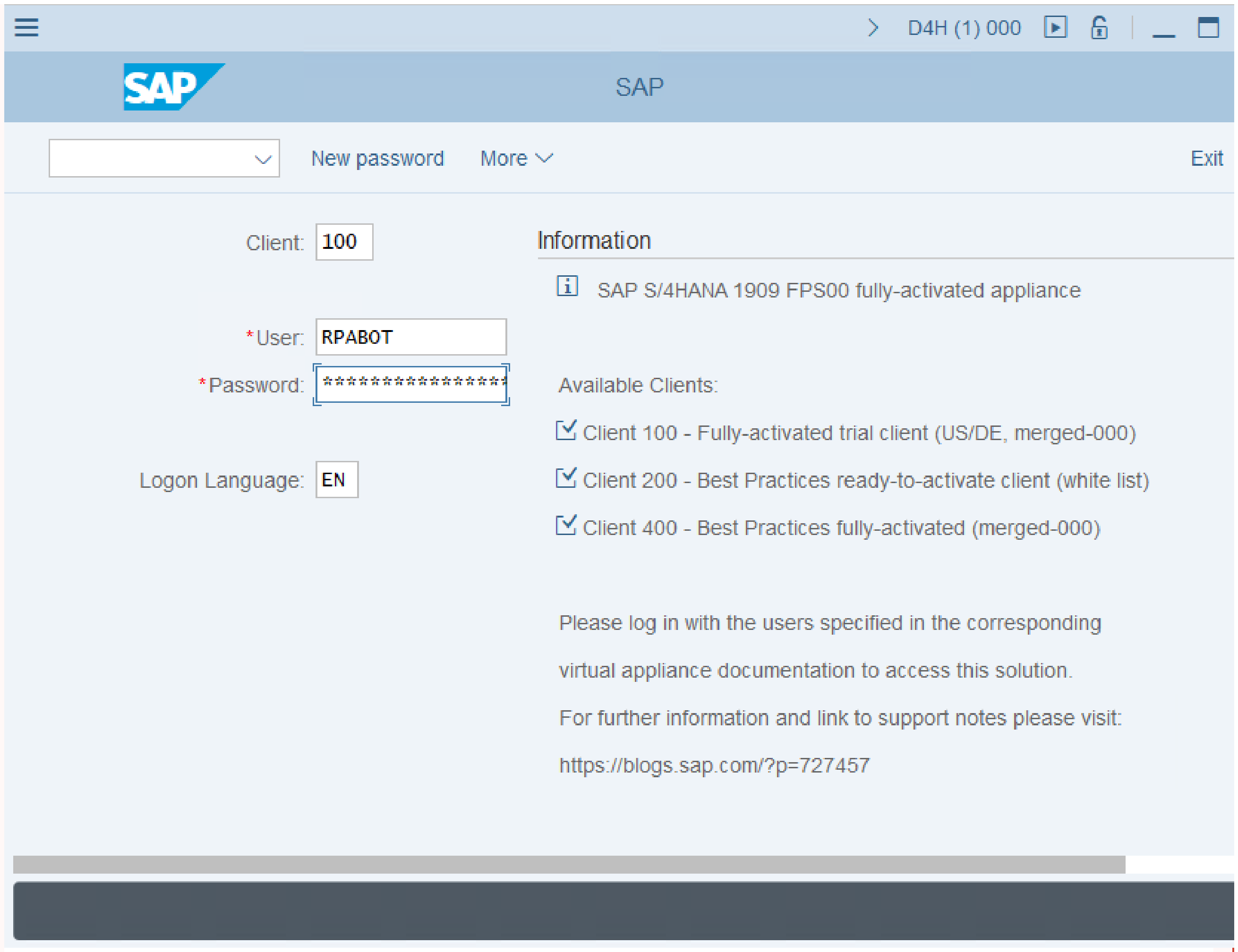
Scripting is installed

User Settings

- [✓] Enable scripting
- [ ] Notify when a script attaches to SAP GUI
- [ ] Notify when a script opens a connection
- [ ] Show native Microsoft Windows dialogs

Enable scripting on server:

Log into created connection using your credentials



Client: 100

*User: RPABOT

*Password: *****************

Logon Language: EN

Information

SAP S/4HANA 1909 FPS00 fully-activated appliance

Available Clients:

Client 100 - Fully-activated trial client (US/DE, merged-000)

Client 200 - Best Practices ready-to-activate client (white list)

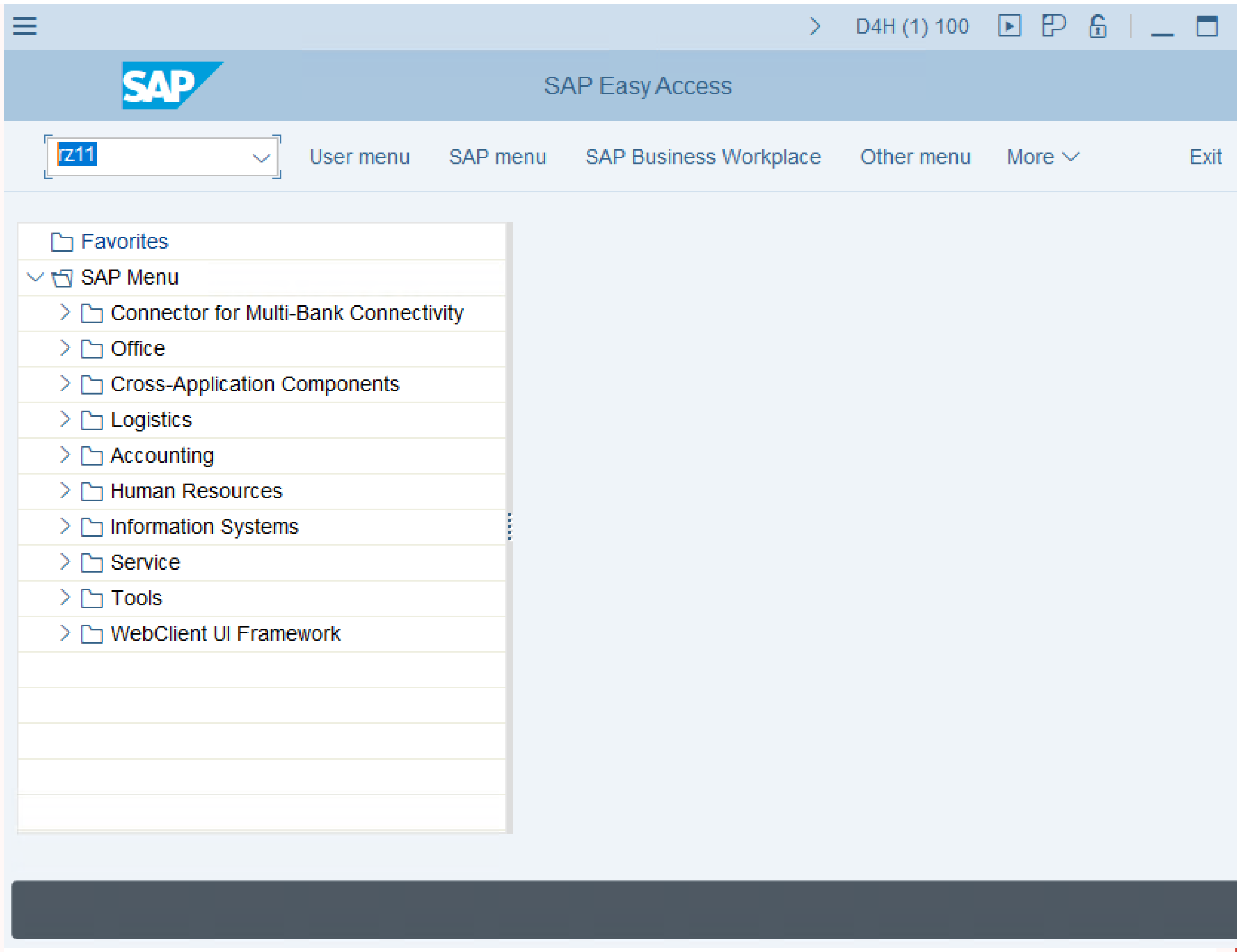
Client 400 - Best Practices fully-activated (merged-000)

Please log in with the users specified in the corresponding virtual appliance documentation to access this solution.

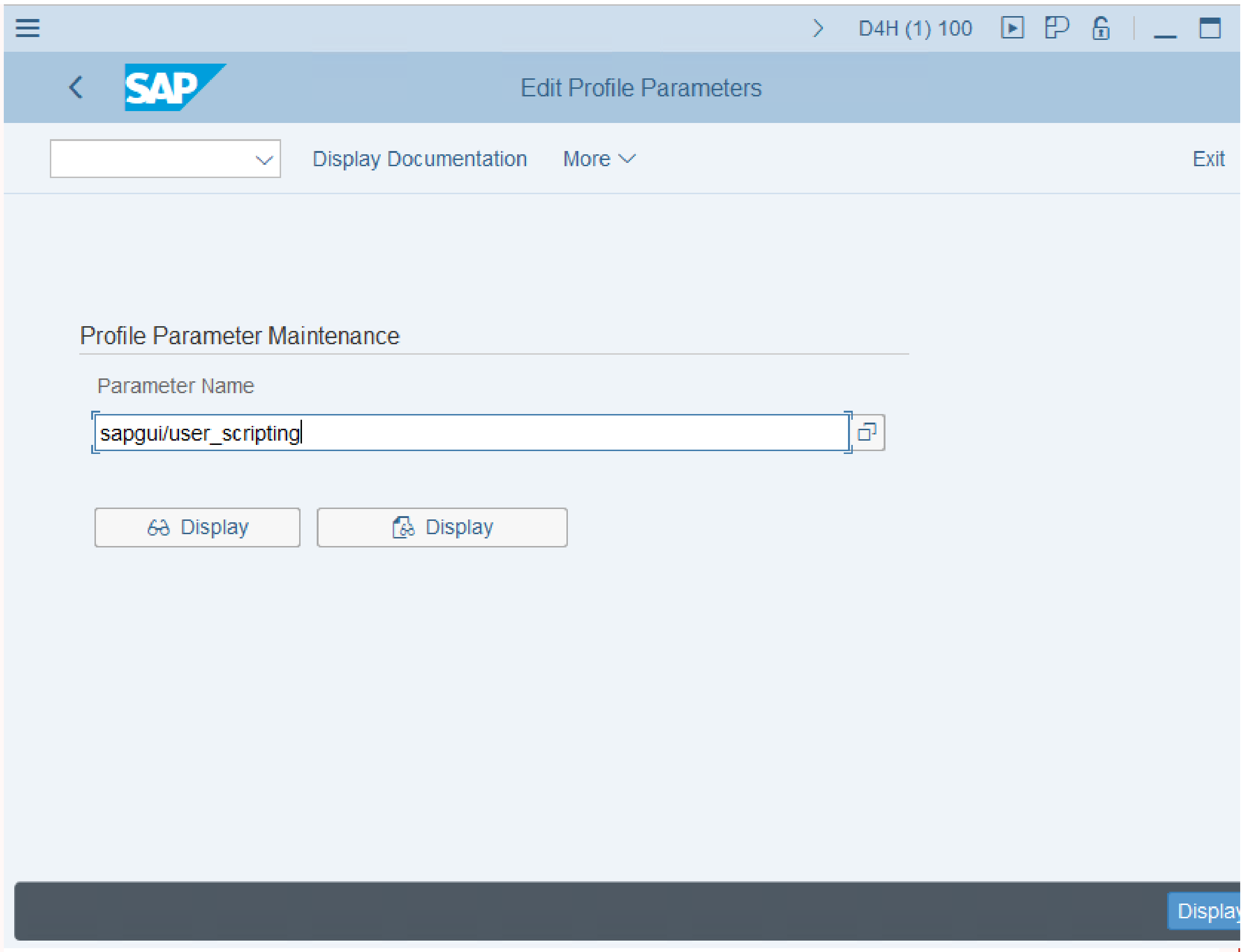
For further information and link to support notes please visit:

https://blogs.sap.com/?p=727457

Run rz11 transaction code



Type in `sapgui/user_scripting` parameter name and press Enter

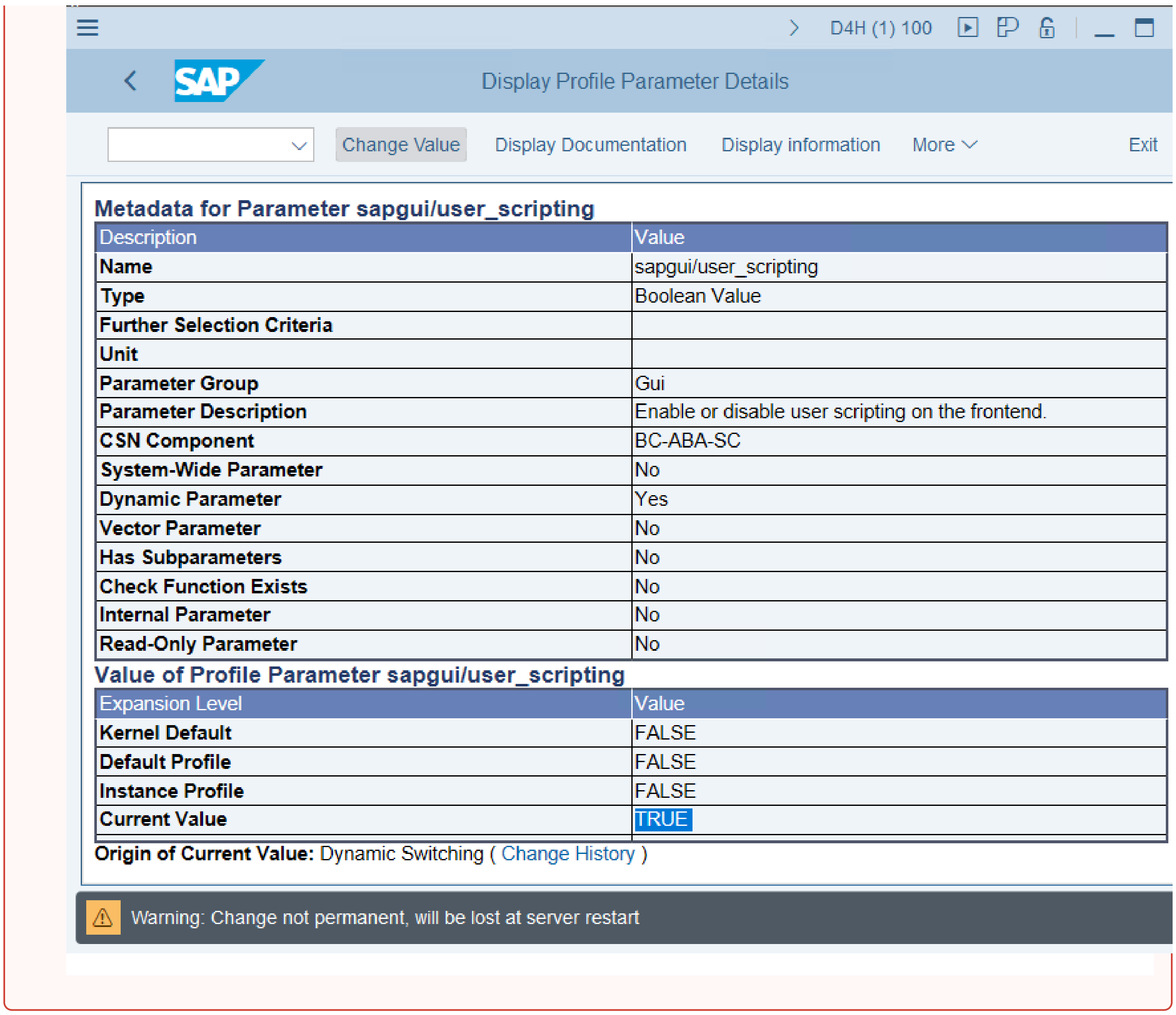


Profile Parameter Maintenance

Parameter Name

sapgui/user_scripting

Make sure current value is set to TRUE, otherwise change it using upper menu


D4H (1) 100

Display Profile Parameter Details

Metadata for Parameter sapgui/user_scripting

Description	Value
Name	sapgui/user_scripting
Type	Boolean Value
Further Selection Criteria	
Unit	
Parameter Group	Gui
Parameter Description	Enable or disable user scripting on the frontend.
CSN Component	BC-ABA-SC
System-Wide Parameter	No
Dynamic Parameter	Yes
Vector Parameter	No
Has Subparameters	No
Check Function Exists	No
Internal Parameter	No
Read-Only Parameter	No

Value of Profile Parameter sapgui/user_scripting

Expansion Level	Value
Kernel Default	FALSE
Default Profile	FALSE
Instance Profile	FALSE
Current Value	TRUE

Origin of Current Value: Dynamic Switching (Change History)

Warning: Change not permanent, will be lost at server restart

For quick reference please check short SAP example

Driver Initialization

You can initialize the driver in your automation process by the following way:

```
@Driver(Type.SAP)
private SapDriver sapDriver;
```

Opening and closing SAP application

To use SAP driver you have to start SAP session first. To do so you can either employ desktop driver or make use of sapshcut utility bundled with SAP Logon distribution.

Open SAP using sapshcut utility

1. Run `sapshcut` command with all the necessary arguments such as user and password, SID and client and others

```
String sapPath = "C:/Program Files (x86)/SAP/FrontEnd/SAPgui/sapshcut";
String user = "userid";
String password = "password";
String system = "S4H";
String client = "100";

String command = String.format("\"%s\" -user=%s -pw=%s -language=EN -system=%s -client=%s -command=%s -
trace=0",
                sapPath, user, password, system, client);

Runtime.getRuntime().exec(command);
```

Run `sapshcut -help` in command line to find out more avaiable arguments

2. After a certain time when SAP Frontend window appears the SapDriver session can be safely initialized

```
sapDriver.sleep(6000); // sleep until SAP Frontend window shows up
sapDriver.initSession();
```

Open SAP using DesktopDriver

1. Start SAP Logon using desktop driver.

```
@Driver(Type.DESKTOP)
private DesktopDriver desktopDriver;
...

desktopDriver.get("C:/Program Files (x86)/SAP/FrontEnd/SAPgui/saplogon.exe");
```

2. Select the required connection from list and run it.

```
desktopDriver.findElement(By.name(sapConnectionName)).click();
desktopDriver.findElement(By.name(sapConnectionName)).sendKeys(ENTER);
```

3. SAP Frontend window will open which is a subject to SAP driver automation.

```
@Driver(Type.SAP)
private SapDriver sapDriver;

sapDriver.initSession();
```

4. Log into SAP Frontend with required credentials and perform other AP related actions using SAP Driver.

```
sapDriver.findElementById("wnd[0]/usr/txtRSYST-BNAME").setText(user);
sapDriver.findElementById("wnd[0]/usr/pwdRSYST-BCODE").setText(password);
sapDriver.findElementById("wnd[0]").sendKeys(SapKeyCode.ENTER);

// other actions after user has been authenticated
```

5. Logout from SAP Frontend with SAP Driver

```
// logs out without prompt using command input
sapDriver.findElementById("wnd[0]/tbar[0]/okcd").setText("/nex");
```

Close SAP

Before closing SAP window it is important to quit i.e. destroy SapDriver to prevent memory leaks or other troubles associated with underlying system components

```
sapDriver.quit();
```

The remaining SAP window can be closed by killing saplogon process

```
Runtime.getRuntime().exec("TASKKILL /F /IM saplogon.exe /T");
```

If started with DesktopDriver, SAP can also be closed by sending Alt+F4 keys

```
desktopDriver.getKeyboard().sendKeys(Keys.chord(Key.ALT, Key.F4));
```

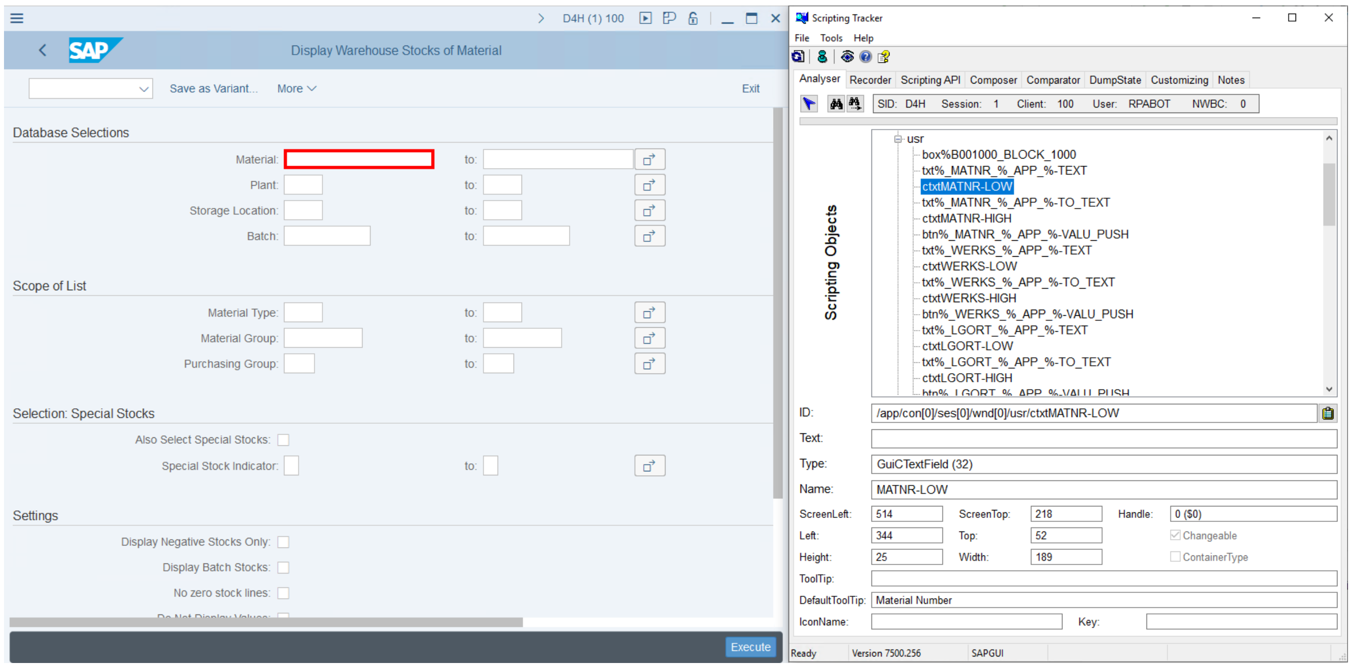
Working with UI Elements

Inspector

To define SAP Element ID run Scripting Tracker and refresh the object tree. Next click on target element using arrow button and copy value from ID field.

Below is the example of retreiving id for Material input field.

```
@FindBy(xpath = "/app/con[0]/ses[0]/wnd[0]/usr/ctxtMATNR-LOW")
private SAPElement materialField;
```



SID: D4H Session: 1 Client: 100 User: RPABOT NWBC: 0

```
usr
  box%B001000_BLOCK_1000
  txt%_MATNR_%_APP_%-TEXT
  ctxtMATNR-LOW
  txt%_MATNR_%_APP_%-TO_TEXT
  ctxtMATNR-HIGH
  btn%_MATNR_%_APP_%-VALU_PUSH
  txt%_WERKS_%_APP_%-TEXT
  ctxtWERKS-LOW
  txt%_WERKS_%_APP_%-TO_TEXT
  ctxtWERKS-HIGH
  btn%_WERKS_%_APP_%-VALU_PUSH
  txt%_LGORT_%_APP_%-TEXT
  ctxtLGORT-LOW
  txt%_LGORT_%_APP_%-TO_TEXT
  ctxtLGORT-HIGH
```

ID:	/app/con[0]/ses[0]/wnd[0]/usr/ctxtMATNR-LOW				
Text:					
Type:	GuiCTextField (32)				
Name:	MATNR-LOW				
ScreenLeft:	514	ScreenTop:	218	Handle:	0 ($0)
Left:	344	Top:	52	✓ Changeable	
Height:	25	Width:	189	ContainerType	
ToolTip:					
DefaultToolTip:	Material Number				
IconName:		Key:			

Find elements

Please, check the table which locators are supported by SapDriver:

Selector type	SapDriver
By.id	
By.name	

By.xpath	
By.ccsSelector	
By.className	
By.tagName	
By.linkText	
By.partialLinkText	
By.desktopSearch	
By.image	
By.anchor	
By.sapId	

The initialized SAP driver can be used to locate page elements.

In order to do so the field ID must be passed to `sapDriver.findElement(By sapId)`.

Now we can access this field inside AP as following:

```
SAPElement descriptionInput = sapDriver.findElement(By.sapId("wnd[0]/usr/txtRSYST-BNAME"));
descriptionInput.setText("EasyRPA material");
```

(v. 2.2) Sap Driver Element

The SapDriver based on ActiveXComponent from the Jacob library (JAVA-COM Bridge), which corresponds to the structure element of the SAP application.

```
SAPElement e1 = getDriver().findElementById("foo");
SAPElement e2 = getDriver().findElementByName("bar");
```

Instance Methods

- ### click

  This emulates manually pressing a button

- ### getDisplayedText

  Returnes DisplayedText propertywhich contains the text as it is displayed on the screen, including preceding or trailing blanks.

- ### sendKeys

  The sendKeys (int keyKode) method takes a code of key or shortcut key as an argument. Performs underlying `setText` call

  ```
  element.sendKeys(SapKeyCode.CTRL_V);
  element.sendKeys(SapKeyCode.ENTER);
  ```

- ### setText

  Sets the value of `text` property.
  The value very much depends on the type of the object on which it is called. This is obvious for text fields or menu items. On the other hand this property is empty for toolbar buttons and is the class id for shells. You can read
  the text property of a label, but you can’t change it, whereas you can only set the text property of a password field, but not read it.

- ### selectDropDownByKey

  Sets the value of the "key" property which is the key of the currently selected item.

- ### getTableRowByIndex

  The method takes the table row index as a parameter and returns SAPElement corresponding to this row
  The indexing supported by this function does not reset the index after scrolling, but counts the rows starting with the first row with respect to the first scroll position. If the selected row is not currently visible then an exception is raised.

  ```
  SapElement row = table.getTableRowByIndex(3);
  ```

- ## checkBoxCheck and checkBoxUncheck

  Changes the `Selected` property, which checks and unchecks the checkbox element.

- ## sendEnter

  Emulates pressing the Enter key.

```
element.sendEnter();

//the same
element.sendKeys(SapKeyCode.ENTER);
```

- ## sendExit

  Emulates pressing the Shift+F3 key shortcut.

```
element.sendExit();

//the same
element.sendKeys(SapKeyCode.SHIFT_F3);
```

(v. 2.2) Sap Driver Window

Current window object can be retrieved from driver using the method:

```
Window window = driver.manage().window();
```

SapWindow supports only the following methods:

Maximize

Maximize the current window.

```
driver.manage().window().maximize();
```

Focus

Calling of this method is focusing on current window.

```
((SapWindow) driver.manage().window()).focus();
```

Get Size

Returns the Dimension object with the "width" and "height" properties

```
Dimension windowDimension = driver.manage().window().getSize();

System.out.println("Current window size is: " + windowDimension.getWidth() + "x" + windowDimension.getHeight());
```

Get Position

Returns the current position of the window on screen. The value is returned as "Point" object with "x" and "y" properties.

```
Point windowPosition = driver.manage().window().getPosition();

System.out.println("Current window position is: " + windowPosition.getX() + "x" + windowPosition.getY());
```

(v. 2.2) Screen Driver

This page describes only Screen Driver specific features and methods which may be different in other drivers.

Implementation which is common for all drivers is described in Drivers page. Please, read it before.

ScreenDriver is used to automate anything on the screen of desktop computer running Windows, Mac or some Linux/Unix.. This driver is based on the **sikuliX** project.

The ScreenDriver class based on the **Screen** class - central object of the sikuli library. According to the documentation, class Screen is there, to have a representation for a pysical monitor where the capturing process (grabbing a rectangle from a screenshot, to be used for further processing with find operations is implemented. The ScreenDriver a set of methods to simplify development, but if you need any specific methods from the Screen class, you can turn to it directly. Below are some examples of initialization and use of the driver.

Driver Initialization

You can initialize the driver in your automation process by the following way:

```
@Driver(DriverParams.Type.SCREEN)
private ScreenDriver screenDriver;
```

Driver Params

Param name	Measure	Default Value	Description
DriverParams. Screen.**MIN_SIMILARITY**	float	0.7	The default minimum similiarty of find operations. While using a "findElement" operation, if only an image file is provided, Screen Driver searches the region using a default minimum similarity of 0.7.

```
@Driver(value = Type.SAP, param = {
        @DriverParameter(key = DriverParams.Screen.MIN_SIMILARITY, direct = "0.8")
})
private SapDriver sapDriver;
```

Opening new applications

Use DesktopDriver if you need to start application using driver. Here the link of how to open new application using DesktopDriver

Window management

There is **no such entity as "Window" in ScreenDriver**, as this driver works with the whole screen like one window and driver can't distinguish between all the different windows on the screen. If you need to perform any window-specific operation (closing, maximizing, etc) you can use a hot-keys.

Working with UI Elements

Prepare screenshots

Screenshots should be taken on the screen where the process is supposed to work. Differences in screen resolution or color scheme may break the process.

1. Take a screenshot of the necessary element and save it to a file.

   We recommend to use an utility that makes it easy to take screenshots. lightshot, for example.

2. Place this file in project resources.
3. You can use the name of image file (or the path to file if it is not in resources) as a selector.

Find elements

Please, check the table which locators are supported by BrowserDriver:

Selector type	BrowserDriver
By.id	
By.name	
By.xpath	
By.ccsSelector	
By.className	
By.tagName	
By.linkText	
By.partialLinkText	
By.desktopSearch	
By.image	
By.anchor	
By.sapId	

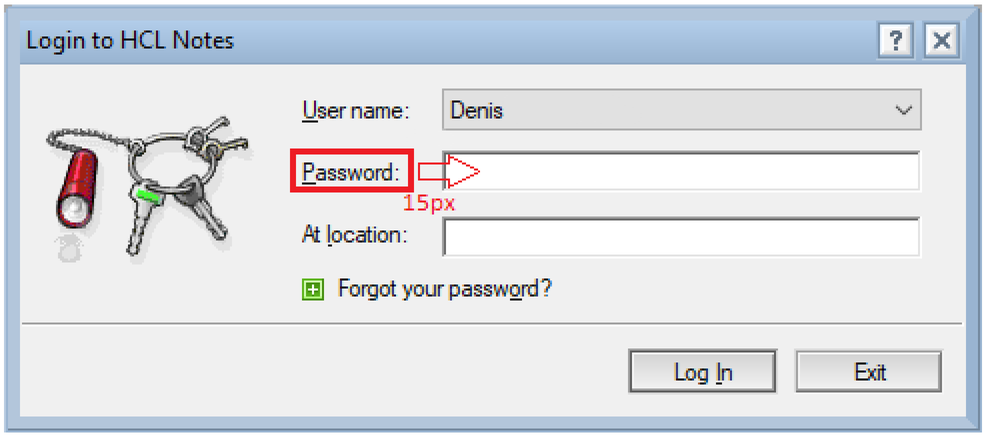
ScreenDriver supports two types of "By" selector: "**By.image**" and "**By.anchor**".

Bellow an example of how to find element using "**By.image**":

```
ImageElement imageElement = driver.findElement(By.image("navigation-btn-1.png"));
```

ScreenDriver also supports "**By.anchor**" selector. It's useful when you can't specify an unique UI area of your target element (because it contains the dynamic text or it's an empty input field, for example), but you can specify some UI area near your target area.

Let's consider the following example, when you need to find a location of "Password" input field:



You can't use input field area as a selector, because there is 2 similar empty inputs: "Password" and "At location".

So you can use the area with "Password:" label (which is highlighted in red) as an **anchor** for password input field. As you know that input follows the label after 15px, your anchor selector will looks like:

```
ImageElement passwordLabelAnchor = driver.findElement(By.image("password-label.png"));
int passwordFieldWidth = 250;
int passwordFieldHeight = 20;

ImageElement passwordInputField = driver.findElement(By.anchor(passwordLabelAnchor, 15, 0, passwordFieldHeight,
passwordFieldWidth));
```

(v. 2.2) Screen Driver Element

ImageElement is a certain rectangular area of the screen. It is possible to search for and get this element only by image (see Find Elements section on drive page):

Instance Methods

- ### contains

    This method accept another "ImageElement" object and return "true" of current element contains another image element inside.

- ### copyToClipboard

    Clicks inside target image element, presses "Ctrl+A" (select all) keys combination and then "Ctrl+C" (copy) keys combination to save selected text into buffer.

- ### doubleClick

    Performs double click on the center of target image element.

- ### getOcrText

    OCR module recognizes and returns a text from target image element.

- ### getRegion

    Returns `org.sikuli.script.Region` object which contains a lot of useful methods.
    For example. region's methods `getX()`, `getY()`, `getW()` and `getH()` identify the position on screen and the element size.

```
System.out.println("x: " + element.getRegion().getX());
System.out.println("y: " + element.getRegion().getY());
System.out.println("width: " + element.getRegion().getW());
System.out.println("height: " + element.getRegion().getH());

//Example output
//x: 447
//y: 826
//width: 70
//height: 47
```

- ### getUrl

    Returns the relative path to the image you are searching by.

- ### highlight

Highlights the region for some period of time. Accepts the time in seconds as double.

- ## rightClick

  Performs right click on the center of target image element.

- ## keyDown

  Presses down and holds the key. Accepts keycode from `org.sikuli.script.Key`

```
element.keyDown(Key.ALT);
element.keyDown(Key.F4);
element.keyUp();
```

- ## keyUp

  Releases all the pressed keys. Can also accept a keycode from `org.sikuli.script.Key` if the specific key must be released

- ## pasteFromClipboard

  Clicks inside target image element and presses "Ctrl+V" (paste) keys combination.

- ## sendEnter

  Sends enter key inside target image element.

(v. 2.2) Data Services

This section describes configuration and data store services the platform provides.

(v. 2.2) Configuration Service

Overview

ConfigurationService provides read access to key-value properties (e.g. URL to resource, Vault alias, Datastore name etc.) which can be used in AP execution and can be changed at any time without redeploy AP.

Define properties in standalone and developer configurations

For those run configurations, properties can be stored in ***resources/apm_run.properties*** file as key-value pairs

Example of ***apm_run.properties***

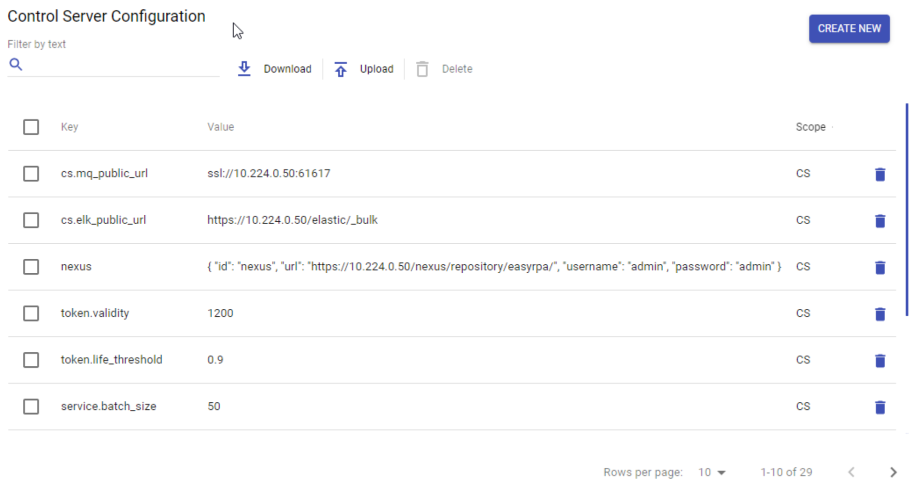
apm_run.properties

```
hello=Hello property
boolean.value=true
url.value=https://172.20.194.57:8444
number.value=123
long.value=9223372036854775807
```

Define properties for Control Server run

There are 3 points, where you can define configuration properties:

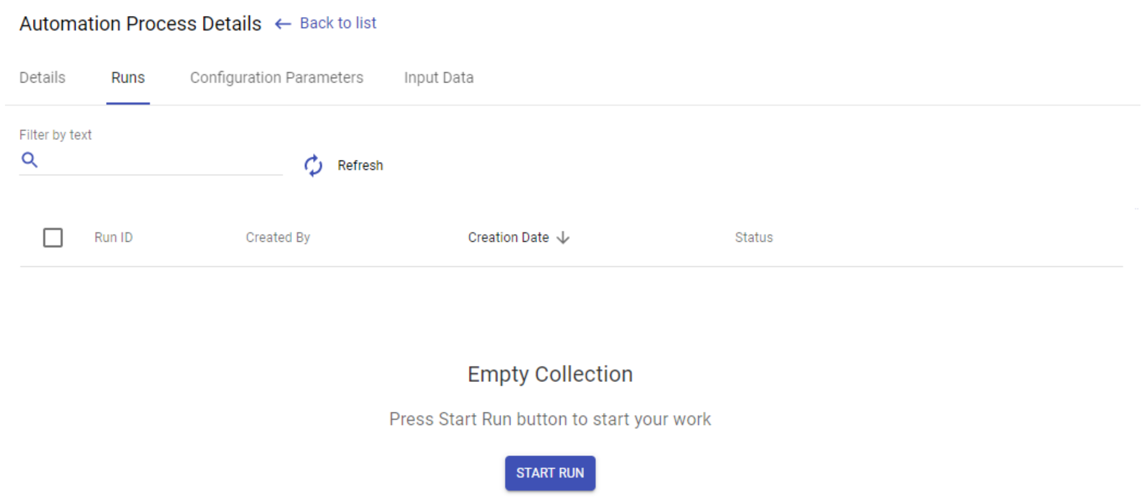
1. Control Server Configuration properties - are visible across all Automation Processes defined in Control Server



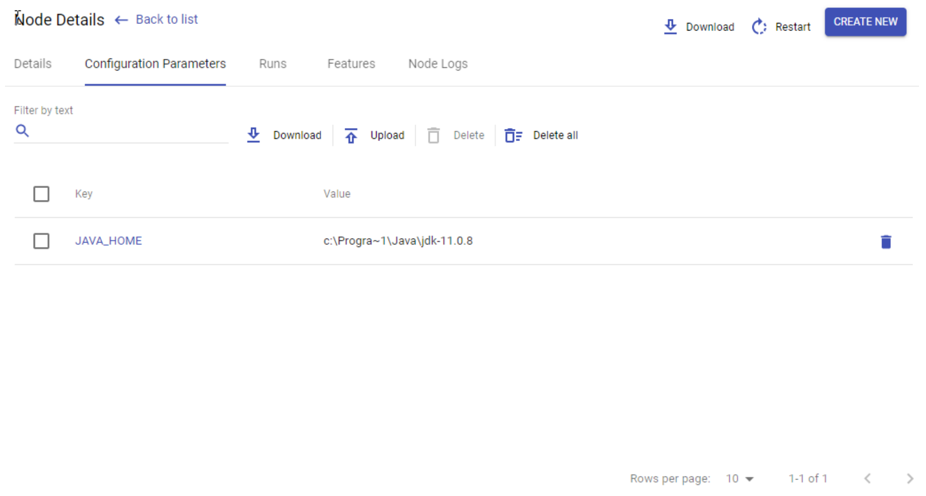
Key	Value	Scope
cs.mq_public_url	ssl://10.224.0.50:61617	CS
cs.elk_public_url	https://10.224.0.50/elastic/_bulk	CS
nexus	{ "id": "nexus", "url": "https://10.224.0.50/nexus/repository/easyrpa/", "username": "admin", "password": "admin" }	CS
token.validity	1200	CS
token.life_threshold	0.9	CS
service.batch_size	50	CS

Rows per page: 10 1-10 of 29

2. Automation Process Configuration properties - are visible only for runs of the Automation Process and overrides properties defined on Control Server Level



3. Node Configuration properties - are applying for runs that is running on the node and overrides properties defined on levels above



Key	Value
JAVA_HOME	c:\Progra~1\Java\jdk-11.0.8

4. Feature related properties on node - will be only injected by feature if there is no properties defined on the level above. This properties are related to driver endpoints, like SELENIUM_HUB_URL, WINDOWS_APPLICATION_DRIVER_URL

When you deploy Automation Process on node, you should define all your required properties in the Control Server properties, the ***resources/apm_run.properties*** is not avalible during runs on node.

Node configuration Properties

There are some parameters, specifying which for node you can effect JVM for Automation Process:

Parameter	Description	Example
JAVA_HOME	Change JVM that will be using for Automation Processes run.	`c:\Program Files\Java\jdk-11.0.8`

JAVA_OPTS	JVM options that will be passed to Automation Process run.
	
	Using it, you can start Automation Process in debug mode.
		`-DmyProperty1=123 -DmyProperty2=123`
		
		`-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=5005`

Configuration Service API

First, you need to *Inject* ConfigurationService in your Task class:

```
@ApTaskEntry(name = "ConfigurationService Example Task")
@Slf4j
public class ConfigurationExample extends ApTask {
    @Inject
    private ConfigurationService cfg;
}
```

Retrieving string using ConfigurationService :

```
String testProperty = this.cfg.get("hello");
```

You can set default value if key does not exist in service:

```
String property2 = this.cfg.get("no.value", "Default value");
```

You can also use predefined formatters to format values:

```
Boolean booleanFromCfg = this.cfg.get("boolean", ConfigurationService.Formatter.BOOLEAN);
Integer number = this.cfg.get("number", ConfigurationService.Formatter.INT);
Long longNumber = this.cfg.get("long.number", ConfigurationService.Formatter.LONG);
URL url = this.cfg.get("url", ConfigurationService.Formatter.URL);
```

Full example:

ConfigurationExample.java

```
package eu.ibagroup.easyrpa.configuration.task;

import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.engine.service.ConfigurationService;
import lombok.extern.slf4j.Slf4j;

import javax.inject.Inject;
import java.net.URL;

@ApTaskEntry(name = "Configuration Service Example")
@Slf4j
public class ConfigurationExample extends ApTask {

    @Inject
    private ConfigurationService cfg;

    @Override
    public void execute() {
        String testProperty = this.cfg.get("hello");
        log.info("hello = {}", testProperty);

        String property2 = this.cfg.get("no.value", "Default value");
        log.info("no.value = {}", property2);

        Boolean booleanFromCfg = this.cfg.get("boolean.value", ConfigurationService.Formatter.BOOLEAN);
        Integer number = this.cfg.get("number.value", ConfigurationService.Formatter.INT);
        Long longNumber = this.cfg.get("long.value", ConfigurationService.Formatter.LONG);
        URL url = this.cfg.get("url.value", ConfigurationService.Formatter.URL);
        log.info("boolean.value = {}", booleanFromCfg);
        log.info("number.value = {}", number);
        log.info("long.value = {}", longNumber);
        log.info("url.value = {}", url);
    }
}
```

ConfigurationExampleAp.java

```
package eu.ibagroup.easyrpa.configuration;

import eu.ibagroup.easyrpa.configuration.task.ConfigurationExample;
import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModuleBase;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import eu.ibagroup.easyrpa.engine.boot.ApModuleRunner;
import lombok.extern.slf4j.Slf4j;

@Slf4j
@ApTaskEntry(name = "Configuration Service Example")
public class ConfigurationExampleAp extends ApModuleBase {

    public TaskOutput run() {
        return execute(getInput(), ConfigurationExample.class).get();
    }

        public static void main(String[] args) {
                ApModuleRunner.localLaunch(ConfigurationExampleAp.class);
        }
}
```

(v. 2.2) Data Store

What is the Data Store?

The **Data Store** is a PostgreSQL-based database with built-in ORM and Control Server interface management. Data Store provides a useful interface for working with automation processes.

Most processes are concentrated around a certain entity. At different steps of the process, this entity is supplemented with data, checked, entered in different systems, etc.
For example, when processing payments, you may need to extract data about the payer from some system, check this data, enter payment data into the customer database and send a report. For such a case, it is convenient to create the entity of payment in the code and save/update its fields and statuses in the Data Store at each processing stage. Thus, the process architecture becomes simple and clear, the risk of data loss is minimized, and the process becomes easily scalable and changeable.

In this article we are going to consider interaction with Data Store from Automation Process code, exactly saving and receiving java-objects.

Usage

1. ### Create an entity class.

   As an example, let's overview the Book class below. It is a simple Data Transfer Object (DTO) with some annotations added. First, let's take look at the class annotations.

```
@Entity(value = "crud_example_books")
@NoArgsConstructor
@AllArgsConstructor
@ToString
@Data
public class Book {
...
```

@Entity - This annotation tells that it's an Entity Class which should be save in Data Store. The "*value*" parameter sets the name of the table.

@NoArgsConstructor - is a lombok annotation that creates a constructor without parameters. **A constructor without parameters is required** to work with the DataStore.

@AllArgsConstructor - is a lombok annotation that generates a constructor with 1 parameter for each field in your class.

@ToString - is a lombok annotation that generates an implementation of the toString() method. By default, it'll print your class name, along with each field, in order, separated by commas.

@Data - is a lombok annotation that bundles the features of @ToString, @EqualsAndHashCode, @Getter/@Setter and @RequiredArgsConstructor together.

Field annotations.

```
         ...
        @Id
    @Column("index")
    private String index;
    @Column("name")
    private String name;
         ...
```

@Id - required annotation, which indicates that field is used as unique identifier.
@Column("column_name") - maps the field from entity object to corresponding table column by specifying column name in brackets.

2. ## Create a repository interface that extends the CrudRepository.

   It is necessary to create a repository interface that extends the CrudRepository<T, ID>. There are 5 default methods for working with the repository:

   CrudRepository.class

   ```
   public interface CrudRepository<T, ID> {
       <S extends T> S save(S entity);

       T findById(ID id);

       List<T> findAllById(Collection<ID> ids);

       List<T> findAll();

       void delete(T entity);
   }
   ```

   To use it, you can leave the interface empty.

   Using these 5 methods, you can create additional helper methods, for example, to search by field or to work with object lists, as in the BookRepository interface below.

3. ## Inject the created repository into the task code.

   To use the created repository, you need to inject it in the task class.

   ```
   @ApTaskEntry(name = "Sample Task")
   @Slf4j
   public class YourTask extends ApTaskBase {

       @Inject
       private YourRepository yourRepository;
   ```

4. ## Use it.

   Now you can use the declared repository. You can create, read, update and delete items. SampleTask.java from the example below demonstrates usage of repository basic operations.

Example

Class Book is the entity we'll save in the DataStore.

Book.java

```
import eu.ibagroup.easyrpa.persistence.annotation.Column;
import eu.ibagroup.easyrpa.persistence.annotation.Entity;
import eu.ibagroup.easyrpa.persistence.annotation.Id;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.AllArgsConstructor;
```

```
import lombok.ToString;

@Entity(value = "crud_example_books")
@NoArgsConstructor
@AllArgsConstructor
@ToString
@Data
public class Book {

    @Id
    @Column("index")
    private String index;

    @Column("name")
    private String name;

    @Column("author")
    private String author;

}
```

BookRepository class which allows to interact with our storage

BookRepository.java

```
import eu.ibagroup.easyrpa.crud.entities.Book;
import eu.ibagroup.easyrpa.persistence.CrudRepository;

import java.util.ArrayList;
import java.util.List;

public interface BookRepository extends CrudRepository<Book, String> {

    //Find books by name
    default List<Book> getBookByName(String name) {

        List<Book> result = new ArrayList<>();
        for (Book book : findAll()){
            if (book.getName().equals(name)){
                result.add(book);
            }
        }
        return result;
    }

    //Save list of books
    default void saveBooks(List<Book> books){

        for (Book book : books) {
            save(book);
        }
    }
}
```

SampleTask demonstrates how to save and receive objects from the repository.

SampleTask.java

```
import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApTaskBase;
import eu.ibagroup.easyrpa.crud.BookRepository;
import eu.ibagroup.easyrpa.crud.entities.Book;
import lombok.extern.slf4j.Slf4j;

import javax.inject.Inject;
import java.util.Arrays;
import java.util.List;
```

```java
import java.util.UUID;

@ApTaskEntry(name = "Sample Task")
@Slf4j
public class SampleTask extends ApTaskBase {

    @Inject
    private BookRepository bookRepository;

    @Override
    public void execute() {
        //Create list of books
        Book thinkingInJava = new Book(UUID.randomUUID().toString(), "Thinking in Java", "Bruce Eckel");
        Book cipollino = new Book(UUID.randomUUID().toString(), "Le avventure di Cipollino", "Giovanni
Francesco Rodari");
        Book wadAndPeace = new Book(UUID.randomUUID().toString(), "War and Peace", "Lev Tolstoy");

        List<Book> books = Arrays.asList(thinkingInJava, cipollino, wadAndPeace);

        //Save books to DataStore
        bookRepository.saveBooks(books);

        //Get books from DataStore and display it
        for(Book book : bookRepository.findAll()){
            log.info(book.toString());
        }

        //Find book by name
        for(Book book : bookRepository.getBookByName("Thinking in Java")){
            log.info(book.toString());
        }

        //Clean repository
        for(Book book : bookRepository.findAll()){
            bookRepository.delete(book);
        }
    }
}
```

Automation process to run the task.

EntitySampleAp.java

```java
import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModuleBase;
import eu.ibagroup.easyrpa.engine.apflow.TaskInput;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import eu.ibagroup.easyrpa.engine.boot.ApModuleRunner;
import eu.ibagroup.easyrpa.engine.boot.ConfigurationModule;
import eu.ibagroup.easyrpa.engine.boot.configuration.StandaloneConfigurationModule;
import eu.ibagroup.easyrpa.crud.task.SampleTask;
import lombok.extern.slf4j.Slf4j;

import java.util.UUID;

@Slf4j
@ApModuleEntry(name = "Sample Automation Process of working with entities")
public class EntitySampleAp extends ApModuleBase {

    public TaskOutput run() throws Exception {
        return execute(getInput(), SampleTask.class).get();
    }


        public static void main(String[] args) {
                ApModuleRunner.localLaunch(EntitySampleAp.class);
        }
}
```

(v. 2.2) File Storage

As the file storage with S3 API we use MinIO.

To work with this file storage via S3 API we have a special utility - **StorageManager**.

Injecting Storage Manager

Storage Manager can be injected into your task class:

```
import eu.ibagroup.easyrpa.utils.storage.StorageManager;

@Inject
private StorageManager storageManager;
```

Storage Manager configuration properties

By default S3Manager which is injected uses the following properties keys from configuration:

- **s3.url** - it's configured by default on Control Server. Required to be configured in your local "properties" file.
- **s3.access_key** - it's configured by default on Control Server. Required to be configured in your local "properties" file.
- **s3.secret_key** - it's configured by default on Control Server. Required to be configured in your local "properties" file.
- **s3.bucket** - default bucket to work with. If property is not configured - "**resources**" bucket will be used by default.

Connect to another S3 compatible applications

You can also use S3Manager to work with any other S3 compatible applications. For that you can create your own S3Manager instance using the following constructor:

```
public S3Manager(String s3EndpointUrl, String s3AccessKey, String s3SecretKey, String bucket,
CannedAccessControlList acl)
```

Where **CannedAccessControlList acl** is access control list for object you insert into file storage. This constructor parameter is **optional**. By default **CannedAccessControlList.PublicRead** value is used.

Upload file to S3

To upload the file your can use method with the following signature:

```
public boolean uploadFile(String bucket, String path, InputStream input, String contentType)
```

Where:

- **bucket** - is the target bucket name.
- **path** - is the target file path which includes folders structure, file name and file extension, but without specifying the bucket name. E.g.: "*folder1/subfolder/my_file.txt*"
- **input** - input stream object
- **contentType** - content type of target file. E.g. "*text/plain*". "*image/png*", "*application/pdf*", "*text/html*". By default it's empty.

Returns "true" if the file was successfully uploaded.

The following example demonstrates how to upload file to S3 and get its URL:

```
String fileName = UUID.randomUUID().toString() + ".txt";
String s3FilePath = "upload_file_test/" + fileName;
byte[] fileContent = "Text file content example".getBytes();

boolean fileUploadedSuccessfully = storageManager.uploadFile("bucket", s3FilePath, new ByteArrayInputStream
(fileContent), "text/plain");
if (fileUploadedSuccesfully) {
        String finalS3Url = s3Manager.getS3FileWebUrl(s3FilePath);
        log.info("Test file successfully uploaded. Final URL: {}", finalS3Url);
}
```

Download file from S3

If the file on S3 has public read access rights then it can be easily downloaded by direct URL using the common approach to make HTTP Get request.

But if the file doesn't have public read access rights - it requires S3 credentials for downloading and should be downloaded by S3 API. For that you can use the following method from S3Manager:

```
public InputStream getFile(String bucket, String path)
```

Where:

- **bucket** - is the target bucket name
- **path** - path to target file including folders structure, file name and file extension, but without specifying the bucket name. E.g.: "*folder1/subfolder/my_file.txt*"

The following example demonstrates how to download text file from S3 and read its content:

```
String s3FilePath = "upload_file_test/example.txt";
InputStream fileContentInputStream = s3Manager.getFile("bucket", s3FilePath);
String fileContent = IOUtils.toString(fileContentInputStream, StandardCharsets.UTF_8);
log.info("Test file successfully downloaded. File content: {}", fileContent);
```

Delete file from S3

Method with the following signature should be used to delete file from S3:

```
public boolean deleteFile(String bucket, String path)
```

Where:

- **bucket** - is the target bucket.
- **path** - path to target file including folders structure, file name and file extension, but without specifying the bucket name. E.g.: "*folder1/subfolder/my_file.txt*"

Returns "true" if the file was successfully deleted.

Get list of files

Method with the following signature can be used to get the list of files:

```
public List<String> listFiles(String bucket, String path, String filter)
```

Where:

- **bucket** - is the target bucket.
- **path** - folder of folders structure of the files you're looking for. This parameter is used as prefix of file paths it searches. E.g.: "*folder1/subfolder/*", "*folder2/*".
- **filter** - the Regexp pattern to filter the files. E.g. "*.*\\.txt"*" to find any file with .txt extension.

The following example demonstrates how to get the list of all txt files:

```
String baseFolder = "upload_file_test/";
List<String> filesPathList = s3Manager.listFiles(baseFolder, ".*\\.txt");
for(String filePath: filesPathList) {
    log.info("The following file exist on S3: {}", filePath);
}
```

Complete example

UploadAndReadFilesFromS3.java

```
package eu.ibagroup.easyrpa.s3.task;

import eu.ibagroup.easyrpa.engine.annotation.AfterInit;
import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.utils.storage.S3Manager;
import eu.ibagroup.easyrpa.utils.storage.StorageManager;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.io.IOUtils;

import javax.inject.Inject;

import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.nio.charset.StandardCharsets;
import java.util.List;
import java.util.UUID;

@ApTaskEntry(name = "Upload to S3")
@Slf4j
public class UploadAndReadFilesFromS3 extends ApTask {

        @Inject
        private StorageManager storageManager;

        private String baseFolder;

        private String s3FilePath;

        private byte[] fileContent;

        @AfterInit
        public void init() {
                String fileName = UUID.randomUUID().toString() + ".txt";
                baseFolder = "upload_file_test/";
                s3FilePath = baseFolder + fileName;
                fileContent = "Text file content example".getBytes();
        }

        @Override
        public void execute() throws IOException {
                uploadFile();
                printListFiles();
                readFile();
        }

        private void uploadFile() {
```

```
		boolean fileUploadedSuccessfully = storageManager.uploadFile(S3Manager.DEFAULT_S3_BUCKET,
s3FilePath, new ByteArrayInputStream(fileContent));
		if (fileUploadedSuccessfully) {
			String finalS3Url = storageManager.getFileLink(S3Manager.DEFAULT_S3_BUCKET, s3FilePath);
			log.info("Test file successfully uploaded. Final URL: {}", finalS3Url);
		}
	}

	private void printListFiles() {
		List<String> filesPathList = storageManager.listFiles(S3Manager.DEFAULT_S3_BUCKET, baseFolder,
".*\\.txt");
		for (String filePath : filesPathList) {
			log.info("The following file exist on S3: {}", filePath);
		}
	}

	private void readFile() throws IOException {
		InputStream fileContentInputStream = storageManager.getFile(S3Manager.DEFAULT_S3_BUCKET,
s3FilePath);
		String fileContent = IOUtils.toString(fileContentInputStream, StandardCharsets.UTF_8);
		log.info("Test file successfully downloaded. File content: {}", fileContent);
	}
}
```

(v. 2.2) Notification Service

Overview

NotificationService interface provides access to Notification actions such as sending messages, attachments and templates via channel or just template evaluation.

Interface is located in *eu.ibagroup.easyrpa.engine.service* package.

Channel

Channels are used to determine the recipients of the notification and delivery type.

There are two channel types are supported by current version - Email and Slack. Their config is stored in CS configuration section under *notification.channels.config* key.

```
{
        "email": {
                "default-encoding": "utf-8",
                "host": "mailin.iba",
                "port": "25",
                "protocol": "smtp",
                "mail.smtp.auth": "false",
                "mail.smtp.starttls.enable": "true",
                "mail.smtp.ssl.trust": "mailin.iba",
                "from": "easyrpatest@ibagroup.eu",
                "subject": "EasyRPA Notification", // optional
                "replyTo": "easyrpatest@ibagroup.eu", // optional
                "cc": "easyrpaadmin@ibagroup.eu", // optional
                "bcc": "easyrpadevops@ibagroup.eu", // optional
                "outputContentType": "text" // optional
        },
        "slack": {
                "token": "token-example-2000026465459-2Ksi6raocFxRtqJd6c5H7AF1"
        }
}
```

Email channel configuration

All config parameters are shown above. For example Gmail configuration is described here

The "outputContentType" parameter allows to select the message output format between text and html. By default, the type is determined automatically.

Slack channel configuration

1. Create Slack App and add necessary Bot OAuth Scopes such as `chat:write`. Slack app configuration link - https://api.slack.com/apps
2. Go to *CS Configuration notification.channels.config* and add new config in format - "<*config_name*>" : {"token" : "<*Bot User OAuth Token*>"}
3. Create Slack channel and add your App in it. Copy Channel ID as it will be used further (ID starts from *xoxb-...*).
4. Create notification. Use your *config_name* as Config Name and Channel ID as recipient.

Template

Templates could be used if it is needed to insert some data into message dynamically. Control server support two template engines:

- Thymeleaf
  Currently implemented ThymeleafHtml and ThymeleafText handlers. The key difference between *textual* template mode and the markup one is that in a textual template there are no tags into which to insert logic in the form of attributes, so we have to

rely on other mechanisms. Read more about template modes here. Also useful documentation about Thymeleaf dialect is here.

- FreeMarker
  Apache FreeMarke is a *template engine*: a Java library to generate text output (HTML web pages, e-mails, configuration files, source code, etc.) based on templates and changing data. Templates are written in the FreeMarker Template Language (FTL), which is a simple, specialized language
  Getting started manual is here.

All parameters are passed to the template engine as a *Map<String, ?> params.*

There is an example of FreeMarker template for Debtors sample.

debtors_template.txt

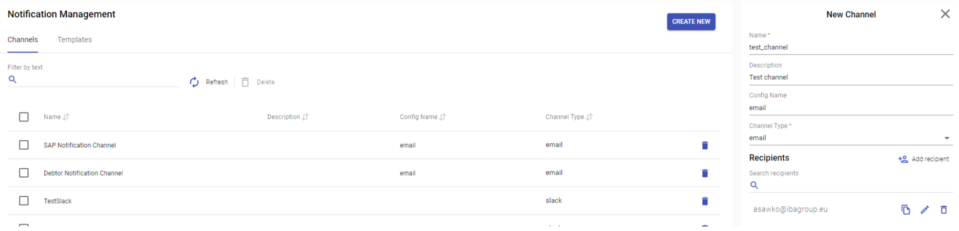
```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <style>
        body {
                background: #fafafa url(https://jackrugile.com/images/misc/noise-diagonal.png);
                color: #444;
                font: 100%/30px 'Helvetica Neue', helvetica, arial, sans-serif;
                text-shadow: 0 1px 0 #fff;
        }

        strong {
                font-weight: bold;
        }

        em {
                font-style: italic;
        }

        table {
                background: #f5f5f5;
                border-collapse: separate;
                box-shadow: inset 0 1px 0 #fff;
                font-size: 14px;
                line-height: 24px;
                margin: 30px auto;
                text-align: center;
                width: 800px;
        }

        caption {
                font-size: 18px;
        font-weight: bold;
        }

        th {
                background: url(https://jackrugile.com/images/misc/noise-diagonal.png), linear-gradient(#777,
#444);
                border-left: 1px solid #555;
                border-right: 1px solid #777;
                border-top: 1px solid #555;
                border-bottom: 1px solid #333;
                box-shadow: inset 0 1px 0 #999;
                color: #fff;
                font-weight: bold;
                padding: 10px 15px;
                position: relative;
                text-shadow: 0 1px 0 #000;
        }

        th:after {
                background: linear-gradient(rgba(255, 255, 255, 0), rgba(255, 255, 255, .08));
                content: '';
                display: block;
                height: 25%;
```

```
		left: 0;
		margin: 1px 0 0 0;
		position: absolute;
		top: 25%;
		width: 100%;
	}

	th:first-child {
		border-left: 1px solid #777;
		box-shadow: inset 1px 1px 0 #999;
	}

	th:last-child {
		box-shadow: inset -1px 1px 0 #999;
	}

	td {
		border-right: 1px solid #fff;
		border-left: 1px solid #e8e8e8;
		border-top: 1px solid #fff;
		border-bottom: 1px solid #e8e8e8;
		padding: 10px 15px;
		position: relative;
		transition: all 300ms;
	}

	td:first-child {
		box-shadow: inset 1px 0 0 #fff;
	}

	td:last-child {
		border-right: 1px solid #e8e8e8;
		box-shadow: inset -1px 0 0 #fff;
	}

	tr {
		background: url(https://jackrugile.com/images/misc/noise-diagonal.png);
	}

	tr:nth-child(odd) td {
		background: #f1f1f1 url(https://jackrugile.com/images/misc/noise-diagonal.png);
	}

	tr:last-of-type td {
		box-shadow: inset 0 -1px 0 #fff;
	}

	tr:last-of-type td:first-child {
		box-shadow: inset 1px -1px 0 #fff;
	}

	tr:last-of-type td:last-child {
		box-shadow: inset -1px -1px 0 #fff;
	}

	tbody:hover td {
		color: transparent;
		text-shadow: 0 0 3px #aaa;
	}

	tbody:hover tr:hover td {
		color: #444;
		text-shadow: 0 1px 0 #fff;
	}

	</style>
</head>
<body>

<div class="container">
	<table class="responsive-table">
```

```
                <caption>   </caption>
                <thead>
                <tr>
                        <th scope="col"></th>
                        <th scope="col"></th>
                        <th scope="col"></th>
                        <th scope="col"></th>
                        <th scope="col"></th>
                        <th scope="col"></th>
                </tr>
                </thead>
                <tfoot>
                <tr>
                        <td colspan="7" align="left">
                                Sources:
                                <#list links?keys as key>
                                        <a href="${links[key]}" rel="external">${key}</a>.
                                </#list>
                                Data is current as of ${date}.
                        </td>
                </tr>
                </tfoot>
                <tbody>
                <#list debtors as debtor>
                        <#if debtor??>
                                <tr>
                                        <td data-title="">${debtor.companyID}</td>
                                        <td data-title="">${debtor.companyName}</td>
                                        <#if debtor.customsFound??>
                                                <td data-title="" data-type="currency">${(debtor.customsFound=="
true") ? then("", "")}</td>
                                        </#if>
                                        <#if debtor.nationalBankFound??>
                                                <td data-title="" data-type="currency">${(debtor.
nationalBankFound=="true") ? then("", "")}</td>
                                        </#if>
                                        <#if debtor.taxesFound??>
                                                <td data-title="" data-type="currency">${(debtor.taxesFound=="
true") ? then("", "")}</td>
                                        </#if>
                                        <#if debtor.socialfundFound??>
                                                <td data-title="" data-type="currency">${(debtor.
socialfundFound=="true") ? then("", "")}</td>
                                        </#if>
                                </tr>
                        </#if>
                </#list>
                </tbody>
        </table>
</div>
</body>
</html>
```

Usage

1. Create Channel in Notification Management section. Config Name should be one of *notification.channels.config* keys.


Notification Management

	Name	Description	Config Name	Channel Type
☐	SAP Notification Channel		email	email
☐	Debtor Notification Channel		email	email
☐	TestSlack			slack

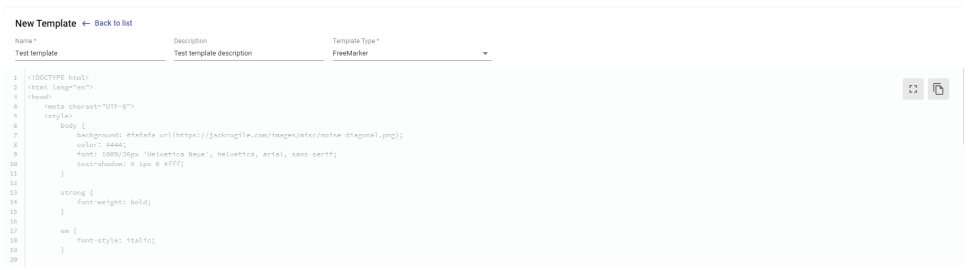
New Channel

Name * test_channel
Description Test channel
Config Name email
Channel Type * email

Recipients

asawko@ibagroup.eu

2. Create Template in Notification Management section. It could be validated by one of the entities - AutomationProcess, DocumentType, Node, Schedule.


New Template ← Back to list

Name * Test template

Description Test template description

Template Type * FreeMarker

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <style>
        body {
            background: #fafafa url(https://jackrugile.com/images/misc/noise-diagonal.png);
            color: #444;
            font: 100%/30px 'Helvetica Neue', helvetica, arial, sans-serif;
            text-shadow: 0 1px 0 #fff;
        }

        strong {
            font-weight: bold;
        }

        em {
            font-style: italic;
        }

```

3. Inject NotificationService in ApTask as well as notification channel and template names.

ApTask configuration

```
@Inject
private NotificationService notificationService;

@Configuration(value = "notificationChannel", defaultValue = "Default Notification Channel")
private String notificationChannel;

@Configuration(value = "notificationTemplate", defaultValue = "Default Email Template")
private String notificationTemplate;
```

4. Use one of the provided interface methods:

NotificationService.java

```
/**
* Evaluates template with provided parameters.
* @param templateName template name
* @param params params for template
* @return the array of byte for processed template
*/
byte[] evaluateTemplate(String templateName, Map<String, ?> params);

/**
* Evaluates template with provided parameters and sends result into specified channel.
* @param templateName template name
* @param params params for template
* @param channelName name of channel for sending
* @param attachments list of attachments for channel if needed
*/
void evaluateTemplateAndSend(String templateName, Map<String, ?> params, String channelName, List<File>
attachments);

/**
* Sends notification into specified channel.
* @param channelName name of channel for sending
* @param text body text to send
* @param attachments list of attachments for channel if needed
*/
void sendNotification(String channelName, String text, List<File> attachments);
```

Examples

- Evaluating and sending template in the Debtors sample.

GenerateEmailReport.java

```
package eu.ibagroup.easyrpa.debtors.task;

import eu.ibagroup.easyrpa.debtors.domain.Debtor;
import eu.ibagroup.easyrpa.debtors.repository.DebtorRepository;
import eu.ibagroup.easyrpa.debtors.to.DebtorsResultTo;
import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.annotation.Configuration;
import eu.ibagroup.easyrpa.engine.annotation.Input;
import eu.ibagroup.easyrpa.engine.annotation.Output;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.engine.service.ConfigurationService;
import eu.ibagroup.easyrpa.engine.service.NotificationService;

import javax.inject.Inject;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@ApTaskEntry(name = "Generate Email Report")
public class GenerateEmailReport extends ApTask {

   @Input(ExtractDebtors.DEBTOR_KEY)
   @Output(ExtractDebtors.DEBTOR_KEY)
   private DebtorsResultTo debtorsResult;

   @Configuration(value = "notificationChannel", defaultValue = "Debtor Notification Channel")
   private String notificationChannel;

   @Configuration(value = "notificationTemplate", defaultValue = "Debtor Email Template")
   private String notificationTemplate;

   @Inject
   private DebtorRepository repo;

   @Inject
   private NotificationService notificationService;

   @Override
   public void execute() {
      try {
         List<Debtor> debtors = repo.fetchDebtors(debtorsResult.getDebtorIds());

         Map data = new HashMap<>();
         data.put("debtors", debtors);
         data.put("date", new SimpleDateFormat("MMMM dd, yyyy").format(new Date()));
         data.put("links", getLinks());

         notificationService.evaluateTemplateAndSend(notificationTemplate, data, notificationChannel,
new ArrayList<>());
         debtorsResult.setGenerateEmailReportComplete(true);
      } catch (Exception e) {
         handleStepError(e);
      }
   }

   @Inject
   private ConfigurationService cfg;

   private Map getLinks() {
      HashMap<String, String> links = new HashMap<>();
      links.put("", cfg.get("gtk.client.url", "http://www.gtk.gov.by/ru/dolg-test-ru/"));
      links.put("", cfg.get("nbrb.client.url", "http://www.nbrb.by/system/banks/guaranteesregister/"));
      links.put("", cfg.get("court.client.url", "http://service.court.by/ru/juridical/judgmentresults
/"));
      links.put("", cfg.get("nalog.client.url", "http://www.portal.nalog.gov.by/debtor/"));
      links.put("", cfg.get("ssf.client.url", "https://ssf.gov.by/ru/debtors-ru/"));
      return links;
   }
}
```

- Sending notification with attachment in the SAP sample.

GenerateAndSendReportTask.java

```
package eu.ibagroup.easyrpa.ap.creatematerial.task;

import eu.ibagroup.easyrpa.ap.creatematerial.ApConstants;
import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.annotation.Configuration;
import eu.ibagroup.easyrpa.engine.annotation.Input;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.engine.service.NotificationService;
import eu.ibagroup.easyrpa.excel.SpreadsheetDocument;
import lombok.extern.slf4j.Slf4j;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;

import javax.inject.Inject;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.Files;
import java.nio.file.StandardCopyOption;
import java.util.*;

import static eu.ibagroup.easyrpa.ap.creatematerial.ApConstants.TEMPLATE_NAME;

@ApTaskEntry(name = "Generate and send report")
@Slf4j
public class GenerateAndSendReportTask extends ApTask {

        @Input
        private String materialCode;

        @Inject
        private NotificationService notificationService;

        @Configuration(value = "notificationTemplate", defaultValue = "SAP Email Template")
        private String notificationTemplate;

        @Configuration(value = "notificationChannel", defaultValue = "SAP Notification Channel")
        private String notificationChannel;

        @Override
        public void execute() {
                try {
                        File attachment = getAttachment();
                        notificationService.evaluateTemplateAndSend(notificationTemplate, new
HashMap<>(),

notificationChannel, Collections.singletonList(attachment));
                        attachment.delete();
                } catch (Exception e) {
                        handleStepError(e);
                }
        }

        private File getAttachment() throws IOException {
                SpreadsheetDocument doc = createSpreadsheet();
                InputStream initialStream = doc.getInputStream();
                File resultFile = new File("execution_results.xlsx");
                Files.copy(
                        initialStream,
                        resultFile.toPath(),
                        StandardCopyOption.REPLACE_EXISTING);
                return resultFile;
        }

        private SpreadsheetDocument createSpreadsheet() {
                SpreadsheetDocument doc = new SpreadsheetDocument(getClass().getResourceAsStream
(TEMPLATE_NAME));
                Sheet sheet = doc.getActiveSheet();
                Row row = sheet.createRow(1);
                row.createCell(0).setCellValue(this.materialCode);
                row.createCell(1).setCellValue(ApConstants.EASYRPA_BASE_UNIT);
                row.createCell(2).setCellValue(String.format(ApConstants.EASYRPA_DESCR_STRING_TEMPL,
this.materialCode));
                return doc;
        }
}
```

(v. 2.2) Vault Service

VaultService bean object allows access to encrypted storage.

To make use of the vault service, the related bean must be injected into the task or page object.

```
@ApTaskEntry(name = "Vault task")
@Slf4j
public class VaultTask extends ApTaskBase {
    @Inject
    VaultService vaultService;
}
```

At the time, vault service provides only getter methods

In **standalone configuration** secrets can be stored in resources/vault.properties file as key-value pairs, where key is secret's alias and value is the value itself encoded as base64 string.

Example of vault.properties entry storing "Hello from secret vault!" string by "my.alias" alias:

vault.properties

```
my.alias=SGVsbG8gZnJvbSBzZWNyZXQgdmF1bHQh
```

Retrieving string using vault service:

```
String myValue = vaultService.getSecret("my.alias", String.class);
```

A typical use case for vault service is to store user credentials. In order to so the value must be provided as json map containig user and password keys e.g. `{"user": admin", "password": "123456"}`. Again, in a form of base64 string:

vault.properties

```
mail.user=eyAidXNlciI6IGFkbWluIiwgInBhc3N3b3JkIjogIjEyMzQ1NiIgfQ
```

Retrieving user credentials:

```
SecretCredentials secrets = vaultService.getSecret("mail.user", SecretCredentials.class);
log.info("'mail.user' user:{} password:{}", secrets.getUser(), secrets.getPassword());
```

In **node configuration** secrets are persisted in HashiCorp vault backed by PostgresSQL database.

In this case secret entries are added manually from secret-vault page of Control Server or using Control Server API

More on working with vault on Control Server in User Guide

Full example:

VaultSampleAp.java

```
package eu.ibagroup.easyrpa.vault;

import eu.ibagroup.easyrpa.engine.apflow.ApModule;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import eu.ibagroup.easyrpa.engine.boot.ApModuleRunner;
import eu.ibagroup.easyrpa.vault.task.VaultTask;

public class VaultSampleAp extends ApModule {
        public static void main(String[] args) {
                ApModuleRunner.localLaunch(VaultSampleAp.class);
        }

        @Override
        public TaskOutput run() throws Exception {
                return execute(getInput(), VaultTask.class).get();
        }
}
```

VaultTask.java

```
package eu.ibagroup.easyrpa.vault.task;

import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApTaskBase;
import eu.ibagroup.easyrpa.engine.model.SecretCredentials;
import eu.ibagroup.easyrpa.engine.service.VaultService;
import lombok.extern.slf4j.Slf4j;

import javax.inject.Inject;

@ApTaskEntry(name = "Vault task")
@Slf4j
public class VaultTask extends ApTaskBase {
    @Inject
    VaultService vaultService;

    @Override
    public void execute() {
        SecretCredentials secrets = vaultService.getSecret("mail.user", SecretCredentials.class);
        log.info("'mail.user' user:{} password:{}", secrets.getUser(), secrets.getPassword());

        String myValue = vaultService.getSecret("my.alias", String.class);
        log.info("my.alias:{}", myValue);
    }
}
```

(v. 2.2) Automation Process Creation

(v. 2.2) RPA by Example

https://git.icdc.io/easyrpa/easy-rpa-samples/kb-short-samples

(v. 2.2) Automating Command Line

Below are 2 ways to run commands using the command line and get the result.

1. Using ProcessBuilder

Run cmd commands

```
String[] commands = {"cmd.exe", "/c", "java -version"};
ProcessBuilder builder = new ProcessBuilder(commands);

//Redirect the process's standard error into standard output
builder.redirectErrorStream(true);
Process process = builder.start();

BufferedReader stdInput = new BufferedReader(new InputStreamReader(process.getInputStream()));

log.info("Here is the output of the command:\n");
String s;
while ((s = stdInput.readLine()) != null) {
        log.info(s);
}
```

2. Using Runtime.getRuntime().exec()

Run cmd commands

```
Runtime runtime = Runtime.getRuntime();
String[] commands = {"cmd.exe", "/c", "ping google.com"};
Process process = runtime.exec(commands);

BufferedReader stdInput = new BufferedReader(new InputStreamReader(process.getInputStream()));
BufferedReader stdError = new BufferedReader(new InputStreamReader(process.getErrorStream()));

// Read the output from the command
log.info("Here is the standard output of the command:\n");
String s;
while ((s = stdInput.readLine()) != null) {
        log.info(s);
}

// Read any errors from the attempted command
log.info("Here is the standard error of the command (if any):\n");
while ((s = stdError.readLine()) != null) {
        log.info(s);
}
```

Full example code:

CmdTask.java

```
package eu.ibagroup.easyrpa.cmd;

import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApTaskBase;
import lombok.extern.slf4j.Slf4j;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
```

```
@ApTaskEntry(name = "Cmd Automation")
@Slf4j
public class CmdTask extends ApTaskBase {

    @Override
    public void execute() {

        try {
            printJavaVersion();
            pingGoogle();
        } catch (Exception e) {
            //do something
        }
    }

    private void printJavaVersion() throws IOException {

        String[] commands = {"cmd.exe", "/c", "java -version"};
        ProcessBuilder builder = new ProcessBuilder(commands);

        //Redirect the process's standard error into standard output
        builder.redirectErrorStream(true);
        Process process = builder.start();

        BufferedReader stdInput = new BufferedReader(new InputStreamReader(process.getInputStream()));

        log.info("Here is the output of the command:\n");
        String s;
        while ((s = stdInput.readLine()) != null) {
            log.info(s);
        }
    }

    private void pingGoogle() throws IOException {

        Runtime runtime = Runtime.getRuntime();
        String[] commands = {"cmd.exe", "/c", "ping google.com"};
        Process process = runtime.exec(commands);

        BufferedReader stdInput = new BufferedReader(new InputStreamReader(process.getInputStream()));
        BufferedReader stdError = new BufferedReader(new InputStreamReader(process.getErrorStream()));

        // Read the output from the command
        log.info("Here is the standard output of the command:\n");
        String s;
        while ((s = stdInput.readLine()) != null) {
            log.info(s);
        }

        // Read any errors from the attempted command
        log.info("Here is the standard error of the command (if any):\n");
        while ((s = stdError.readLine()) != null) {
            log.info(s);
        }

    }
}
```

CmdDemoAp.java

```
package eu.ibagroup.easyrpa.cmd;

import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModuleBase;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import lombok.extern.slf4j.Slf4j;

@Slf4j
@ApModuleEntry(name = "Cmd Automation Demo")
```

```
public class CmdDemoAp extends ApModuleBase {

    public TaskOutput run() throws Exception {
        return execute(getInput(), CmdTask.class).get();
    }
}
```

LocalRunner.java

```
package eu.ibagroup.easyrpa.cmd;

import eu.ibagroup.easyrpa.engine.boot.ApModuleRunner;

public class LocalRunner {
        public static void main(String[] args) {
                ApModuleRunner.localLaunch(CmdDemoAp.class);
        }
}
```

(v. 2.2) Download file

When downloading files from the Internet via RPA, there is often a problem: It is necessary to understand when the download is complete. Here is an example of how such functionality can be implemented.

Below is the code of the waitFile() method from the task class. This method, using the NIO2 API, waits for the file to be created in the directory for downloading. The complete code of the class, as well as other classes required to test its work are given below.

waitFile method

```
private String waitFile(int timeoutSeconds) throws IOException {

        Path directory = Paths.get(filePath);

        try (WatchService service = directory.getFileSystem().newWatchService()) {
                directory.register(service, StandardWatchEventKinds.ENTRY_CREATE);

                long timeout = TimeUnit.NANOSECONDS.convert(timeoutSeconds, TimeUnit.SECONDS);
                while (timeout > 0L) {
                        final long start = System.nanoTime();
                        WatchKey key = service.poll(timeout, TimeUnit.NANOSECONDS);
                        if (key != null) {
                                for (WatchEvent<?> event : key.pollEvents()) {
                                        Path path = (Path) event.context();

                                        if (path.toString().toLowerCase().endsWith(FILE_EXTENSION)) {
                                                return path.getFileName().toString();
                                        }
                                }
                                key.reset();
                                timeout -= System.nanoTime() - start;
                        } else {
                                break;
                        }
                }
        } catch (InterruptedException ignore) {
        }
        return null;
}
```

Using the ChromeOptions class, we pass parameters that allow you to start downloading to the specified directory without a dialog window appearing. An example of interaction with such a window can be found in the Upload file article.

DownloadFile.java

```
package eu.ibagroup.easyrpa.filedownload.task;

import eu.ibagroup.easyrpa.engine.annotation.AfterInit;
import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.annotation.Driver;
import eu.ibagroup.easyrpa.engine.annotation.DriverParameter;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.engine.rpa.driver.BrowserDriver;
import eu.ibagroup.easyrpa.engine.rpa.driver.DriverParams;
import eu.ibagroup.easyrpa.filedownload.WebApplication;
import eu.ibagroup.easyrpa.filedownload.page.MainPage;
import lombok.extern.slf4j.Slf4j;
import org.openqa.selenium.Capabilities;
import org.openqa.selenium.chrome.ChromeOptions;

import java.io.IOException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.StandardWatchEventKinds;
import java.nio.file.WatchEvent;
import java.nio.file.WatchKey;
import java.nio.file.WatchService;
import java.util.HashMap;
import java.util.concurrent.TimeUnit;
import java.util.function.Supplier;

@ApTaskEntry(name = "Download file")
@Slf4j
public class DownloadFile extends ApTask {

        private static final String FILE_EXTENSION = ".bin";

        @Driver(value = DriverParams.Type.BROWSER, param = { @DriverParameter(key = DriverParams.Browser.
SELENIUM_NODE_CAPABILITIES, initializer = ChromeInitializer.class) })
        private BrowserDriver browserDriver;

        private String url;

        private static final String filePath = System.getProperty("user.home") + "\\Downloads"; // hetzner.file.
path

        public static final class ChromeInitializer implements Supplier<Capabilities> {

        @Override
        public Capabilities get() {
                ChromeOptions chromeOptions = new ChromeOptions();

                HashMap<String, Object> chromePrefs = new HashMap<>();
                chromePrefs.put("profile.default_content_settings.popups", 0);
                chromePrefs.put("download.default_directory", filePath);

                chromeOptions.setExperimentalOption("prefs", chromePrefs);
                return chromeOptions;
                }
        }

        @AfterInit
        public void init() {
                url = getConfigurationService().get("hetzner.app.url", "https://speed.hetzner.de/");
        }

        @Override
        public void execute() {

                WebApplication webApplication = new WebApplication(browserDriver);
                MainPage mainPage = webApplication.open(url);

                mainPage.dovnloadTestFile();

                String downloadedFileName = null;

                try {
                        downloadedFileName = waitFile(500);
                } catch (IOException e) {
                        // do something
                }

                if (downloadedFileName != null) {
                        log.info("Downloaded file name: " + downloadedFileName);
                } else {
                        log.info("File wasn't downloaded! ");
                        // throw new EasyRpaException...
                }
        }

        private String waitFile(int timeoutSeconds) throws IOException {

                Path directory = Paths.get(filePath);
                try (WatchService service = directory.getFileSystem().newWatchService()) {
                        directory.register(service, StandardWatchEventKinds.ENTRY_CREATE);

                        long timeout = TimeUnit.NANOSECONDS.convert(timeoutSeconds, TimeUnit.SECONDS);
                        while (timeout > 0L) {
                                final long start = System.nanoTime();
                                WatchKey key = service.poll(timeout, TimeUnit.NANOSECONDS);
                                if (key != null) {
                                        for (WatchEvent<?> event : key.pollEvents()) {
                                                Path path = (Path) event.context();

                                                        if (path.toString().toLowerCase().endsWith
(FILE_EXTENSION)) {
                                                                return path.getFileName().toString();
                                                        }
                                        }
                                        key.reset();
                                        timeout -= System.nanoTime() - start;
                                } else {
                                        break;
                                }
                        }
                } catch (InterruptedException ignore) {
                }
                return null;
        }
}
```

MainPage.java

```
package eu.ibagroup.easyrpa.filedownload.page;

import eu.ibagroup.easyrpa.engine.rpa.page.WebPage;
import eu.ibagroup.easyrpa.engine.rpa.po.annotation.Wait;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;

public class MainPage extends WebPage {

    @FindBy(xpath = "//a[@href='100MB.bin']")
    @Wait(waitFunc = Wait.WaitFunc.VISIBLE)
    private WebElement fileLink100mb;


    public void dovnloadTestFile() {
        fileLink100mb.click();
    }
}
```

DownloadFileAp.java

```
package eu.ibagroup.easyrpa.filedownload;

import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModule;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import eu.ibagroup.easyrpa.filedownload.task.DownloadFile;

@ApModuleEntry(name = "File Download Automation Demo")
public class DownloadFileAp extends ApModule {

        public TaskOutput run() throws Exception {
                return execute(getInput(), DownloadFile.class).get();
        }
}
```

WebApplication.java

```
package eu.ibagroup.easyrpa.filedownload;

import eu.ibagroup.easyrpa.engine.rpa.Application;
import eu.ibagroup.easyrpa.engine.rpa.driver.BrowserDriver;
import eu.ibagroup.easyrpa.engine.rpa.element.UiElement;
import eu.ibagroup.easyrpa.filedownload.page.MainPage;

public class WebApplication extends Application<BrowserDriver, UiElement> {

        public WebApplication(BrowserDriver driver) {
                super(driver);
        }

        @Override
        public MainPage open(String... args) {
                String gmailUrl = args[0];
                getDriver().get(gmailUrl);
                return createPage(MainPage.class);
        }
}
```

LocalRunner.java

```
package eu.ibagroup.easyrpa.filedownload;

import eu.ibagroup.easyrpa.engine.boot.ApModuleRunner;

public class LocalRunner {

        public static void main(String[] args) {
                ApModuleRunner.localLaunch(DownloadFileAp.class);
        }
}
```

(v. 2.2) Notepad automation

This example shows how to automate a Windows application using the Desktop driver.

The example consists of several parts:

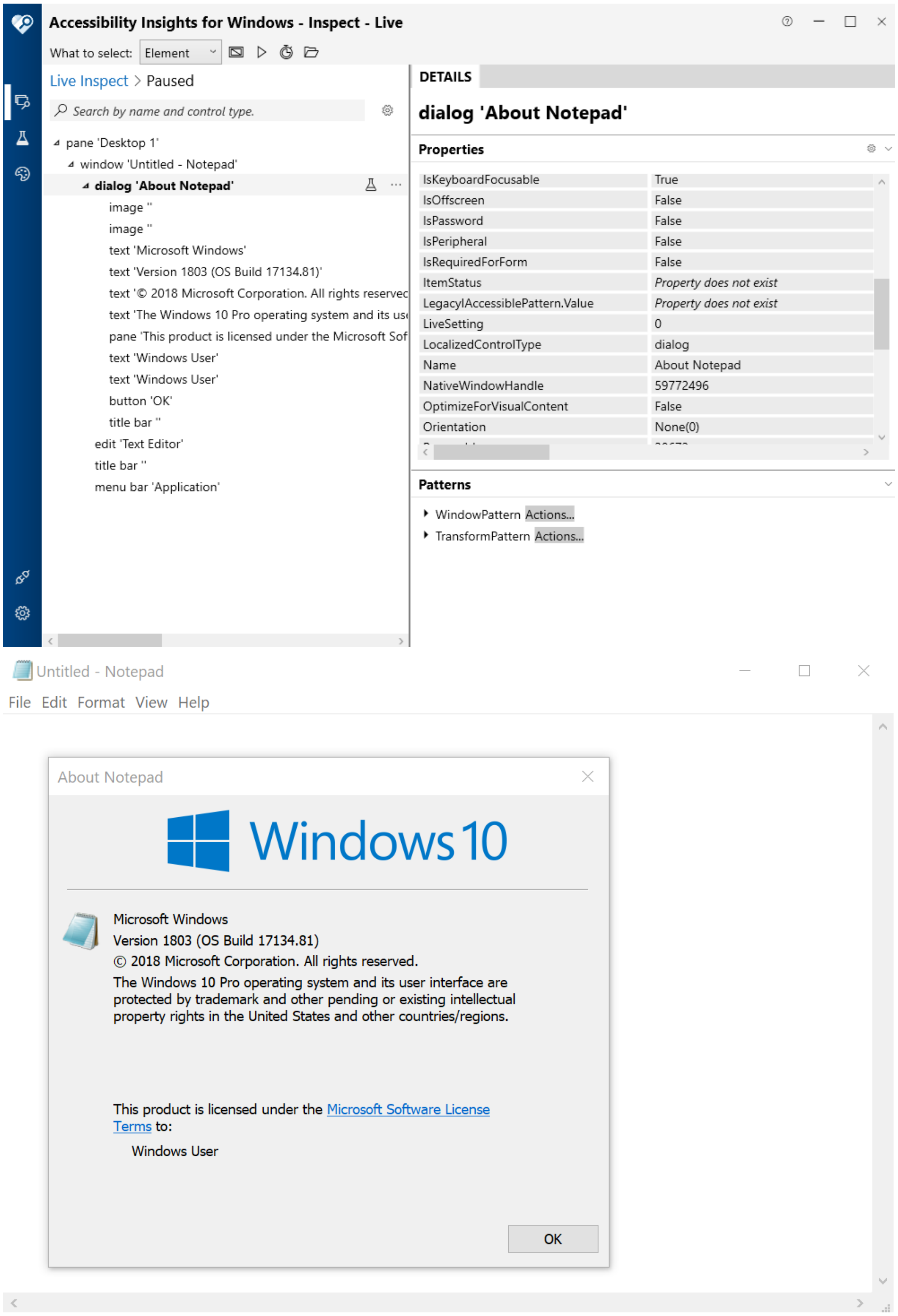
1. Opening a Notepad application
2. Opening the About Notepad RPA window
3. Scrapping application information
4. Saving the information received in Notepad
5. Saving file - interaction with the Save As window

Application class opens Notepad and returns MainPage object.

NotepadApplication.java

```
package eu.ibagroup.easyrpa.notepad;

import eu.ibagroup.easyrpa.engine.rpa.Application;
import eu.ibagroup.easyrpa.engine.rpa.driver.DesktopDriver;
import eu.ibagroup.easyrpa.engine.rpa.element.DesktopElement;
import eu.ibagroup.easyrpa.notepad.page.MainPage;

public class NotepadApplication extends Application<DesktopDriver, DesktopElement> {

        public NotepadApplication(DesktopDriver driver) {
                super(driver);
        }

        @Override
        public MainPage open(String... args) {
                String appPath = args[0];
                getDriver().get(appPath);
                return createPage(MainPage.class);
        }
}
```

As you can see, the "About Notepad" window is an internal element of the main application window. The same goes for the "Save As" window, so we do not have to switch between the windows.



Accessibility Insights for Windows - Inspect - Live
What to select: Element
Live Inspect > Paused
Search by name and control type.
DETAILS

```
pane 'Desktop 1'
  window 'Untitled - Notepad'
    dialog 'About Notepad'
      image ''
      image ''
      text 'Microsoft Windows'
      text 'Version 1803 (OS Build 17134.81)'
      text '© 2018 Microsoft Corporation. All rights reservec
      text 'The Windows 10 Pro operating system and its us
      pane 'This product is licensed under the Microsoft Sof
      text 'Windows User'
      text 'Windows User'
      button 'OK'
      title bar ''
    edit 'Text Editor'
    title bar ''
    menu bar 'Application'
```

dialog 'About Notepad'

Properties

IsKeyboardFocusable	True
IsOffscreen	False
IsPassword	False
IsPeripheral	False
IsRequiredForForm	False
ItemStatus	*Property does not exist*
LegacyAccessiblePattern.Value	*Property does not exist*
LiveSetting	0
LocalizedControlType	dialog
Name	About Notepad
NativeWindowHandle	59772496
OptimizeForVisualContent	False
Orientation	None(0)

Patterns

- WindowPattern Actions...
- TransformPattern Actions...

Microsoft Windows
Version 1803 (OS Build 17134.81)
© 2018 Microsoft Corporation. All rights reserved.
The Windows 10 Pro operating system and its user interface are protected by trademark and other pending or existing intellectual property rights in the United States and other countries/regions.

This product is licensed under the Microsoft Software License Terms to:

Windows User

The MainPage class contains methods for working with the main application window: typing text, opening the "About Notepad" window, saving a document, closing the application.

MainPage.java

```java
package eu.ibagroup.easyrpa.notepad.page;

import eu.ibagroup.easyrpa.engine.annotation.AfterInit;
import eu.ibagroup.easyrpa.engine.rpa.element.DesktopElement;
import eu.ibagroup.easyrpa.engine.rpa.locator.By;
import eu.ibagroup.easyrpa.engine.rpa.locator.UIQuery;
import eu.ibagroup.easyrpa.engine.rpa.page.DesktopPage;
import eu.ibagroup.easyrpa.engine.rpa.po.annotation.WithTimeout;
import org.apache.commons.io.FilenameUtils;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.Keys;

import java.io.File;

public class MainPage extends DesktopPage {

        @FindBy(name = "Help")
        @WithTimeout(time = 3)
        private DesktopElement helpMenu;

        @FindBy(name = "About Notepad")
        @WithTimeout(time = 3)
        private DesktopElement aboutNotepadMenuItem;

        private String title;

        public AboutNotepadPage openAboutWindow() {
                helpMenu.click();
                aboutNotepadMenuItem.click();
                return createPage(AboutNotepadPage.class);
        }

        @AfterInit
        public void init() {
                title = "Untitled - Notepad";
                this.switchToMainWindow();
        }

        public void switchToMainWindow() {
                getDriver().waitAndSwitchToWindow(title, 5);
        }

        public void printText(String text) {
                getDriver().findElement(By.desktopSearch(UIQuery.builder().name("Text Editor").build())).
sendKeys(text);
        }

        public void saveAs(File file) {
                getDriver().findElement(By.desktopSearch(UIQuery.builder().name("Text Editor").build())).
click();
                getDriver().getKeyboard().pressKey(Keys.LEFT_CONTROL);
                getDriver().getKeyboard().sendKeys("s");
                getDriver().getKeyboard().releaseKey(Keys.LEFT_CONTROL);
                getDriver().waitAndSwitchToWindow("#32770", "Save As", 5);
                DesktopElement textBox = getDriver().findElement(By.desktopSearch(UIQuery.builder().className
("Edit").name("File name:").build()));
                textBox.clear();
                textBox.sendKeys(file.getAbsolutePath());
                getDriver().findElement(By.desktopSearch(UIQuery.builder().name("Save").build())).click();
                title = FilenameUtils.removeExtension(file.getName()) + " - Notepad";
                switchToMainWindow();
        }

        public void close() {
                switchToMainWindow();
                this.getDriver().close();
        }
}
```

The "About Notepad" window contains several similar elements with text information about the application.
The getContent() method returns a list of strings with text from these elements.
The getContent() method closes the window.

AboutNotepadPage.java

```java
package eu.ibagroup.easyrpa.notepad.page;

import eu.ibagroup.easyrpa.engine.annotation.AfterInit;
import eu.ibagroup.easyrpa.engine.rpa.element.DesktopElement;
import eu.ibagroup.easyrpa.engine.rpa.page.DesktopPage;
import eu.ibagroup.easyrpa.engine.rpa.po.annotation.WithTimeout;
import org.openqa.selenium.support.FindBy;

import java.util.List;
import java.util.stream.Collectors;

public class AboutNotepadPage extends DesktopPage {

        @FindBy(tagName = "Text")
        private List<DesktopElement> textContent;

        @FindBy(tagName = "Button", name = "OK")
        @WithTimeout(time = 3)
        private DesktopElement okBtn;

        @AfterInit
        public void init() {
                getDriver().waitAndSwitchToWindow("About Notepad", 5);
        }

        public List<String> getContent() {
                return this.textContent.subList(0, 1)
                        .stream().map(DesktopElement::getName).collect(Collectors.toList());
        }

        public void close() {
                this.okBtn.click();
        }
}
```

CreateNote - task-class containing the main algorithm of the task.

CreateNote.java

```java
package eu.ibagroup.easyrpa.notepad.task;

import eu.ibagroup.easyrpa.engine.annotation.AfterInit;
import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.annotation.Driver;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.engine.rpa.driver.DesktopDriver;
import eu.ibagroup.easyrpa.engine.rpa.driver.DriverParams;
import eu.ibagroup.easyrpa.notepad.NotepadApplication;
import eu.ibagroup.easyrpa.notepad.page.AboutNotepadPage;
import eu.ibagroup.easyrpa.notepad.page.MainPage;

import java.io.File;
import java.util.List;
import java.util.UUID;

@ApTaskEntry(name = "Get product list from InvoicePlane")
public class CreateNote extends ApTask {
        @Driver(DriverParams.Type.DESKTOP)
        private DesktopDriver desktopDriver;

        private String appPath;

        private String filePath;

        @AfterInit
        public void init() {
                appPath = getConfigurationService().get("notepad.app.path", "C:\\Windows\\system32\\notepad.
exe");
                filePath = getConfigurationService().get("notepad.file.path", System.getProperty("user.home"));
        }

        @Override
        public void execute() {

                NotepadApplication notepadApplication = new NotepadApplication(desktopDriver);
                MainPage mainPage = notepadApplication.open(appPath);
                AboutNotepadPage aboutNotepadPage = mainPage.openAboutWindow();
                List<String> content = aboutNotepadPage.getContent();
                aboutNotepadPage.close();
                mainPage.switchToMainWindow();
                for (String s : content) {
                        mainPage.printText(s + "\n");
                }

                File file = new File(filePath + "\\" + UUID.randomUUID().toString() + ".txt");
                mainPage.saveAs(file);
                mainPage.close();
        }
}
```

Automation process and runner classes.

NotepadAp.java

```
package eu.ibagroup.easyrpa.notepad;

import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModule;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import eu.ibagroup.easyrpa.notepad.task.CreateNote;

@ApModuleEntry(name = "Notepad Automation Demo")
public class NotepadAp extends ApModule {

        public TaskOutput run() throws Exception {
                return execute(getInput(), CreateNote.class).get();
        }
}
```

LocalRunner.java

```
package eu.ibagroup.easyrpa.notepad;

import eu.ibagroup.easyrpa.engine.boot.ApModuleRunner;

public class LocalRunner {

        public static void main(String[] args) {
                ApModuleRunner.localLaunch(NotepadAp.class);
```

```
          }
}
```

(v. 2.2) SAP

- Prepare SAP client connection
- Enable scripting
- Demo overview

Prepare SAP client connection

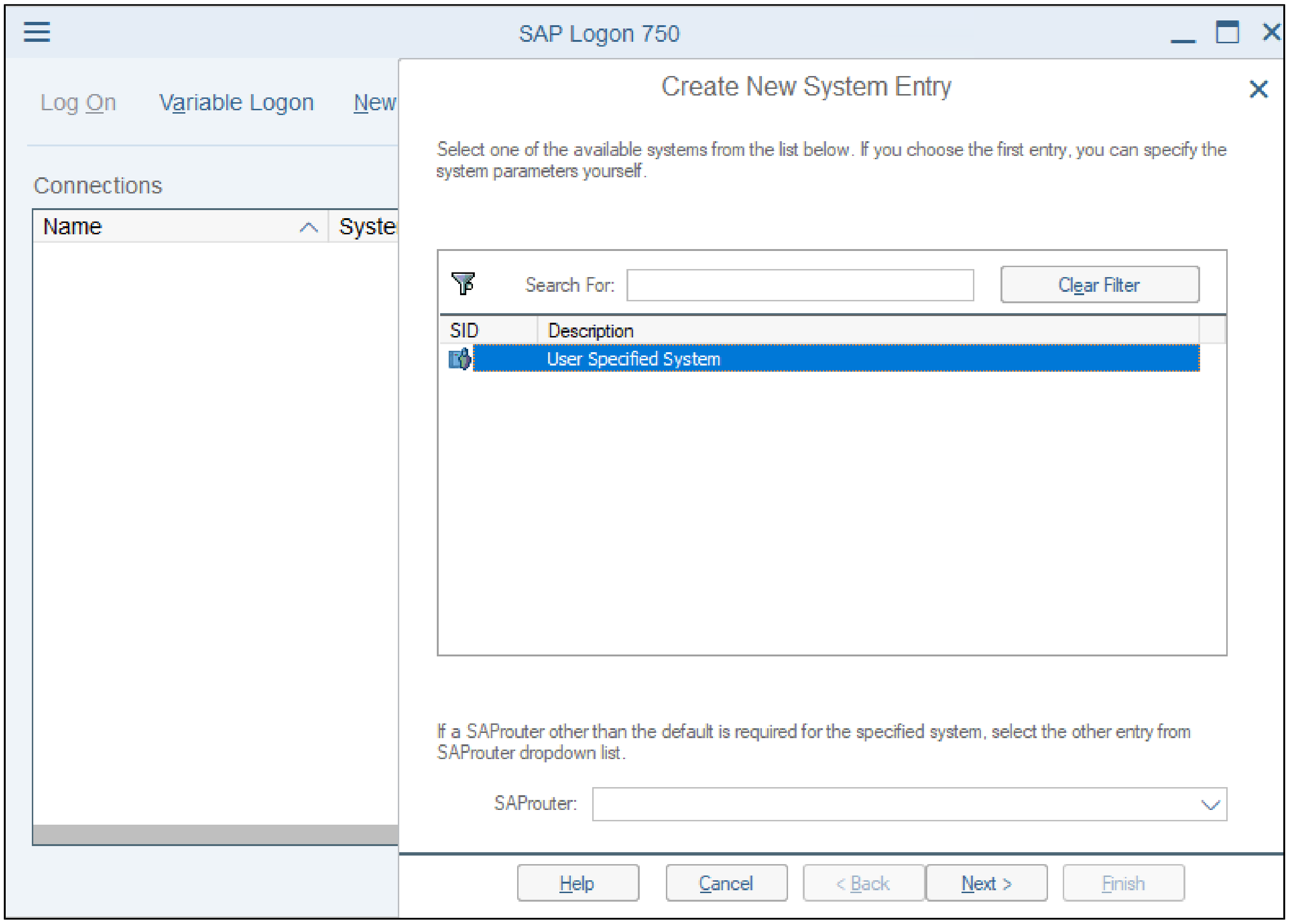
Automation process must utilize WinappDriver on the same machine where NodeAgent is running and SAP environment is configured

SAP main window must hold all the elements without the need to stretch or maximize from standard view

Run Sap Logon and create new connection.

ⓘ The setup stepas will be presented for EasyRPA demo example.

Click new New Item in upper menu:


SAP Logon 750

Create New System Entry

Select one of the available systems from the list below. If you choose the first entry, you can specify the system parameters yourself.

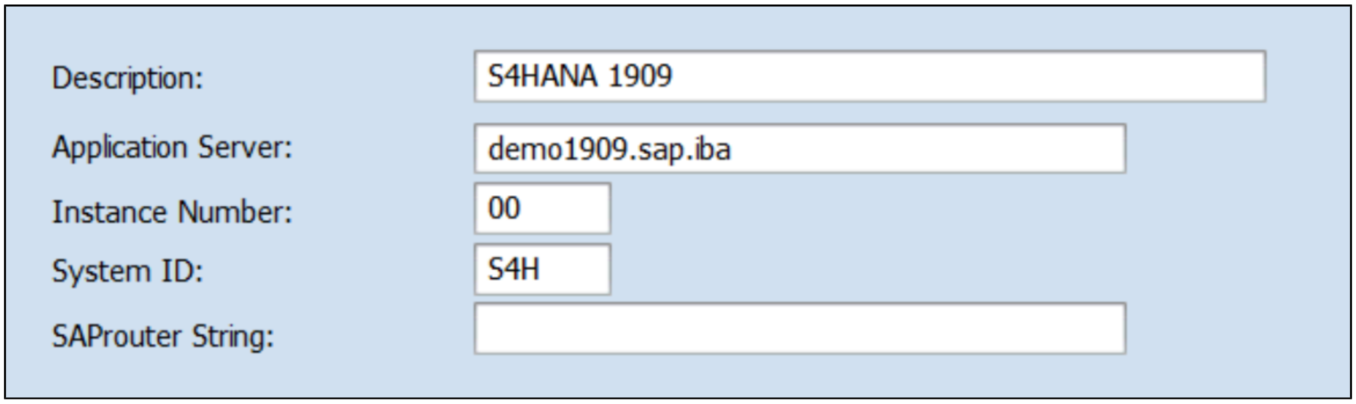
Search For:

SID	Description
	User Specified System

If a SAProuter other than the default is required for the specified system, select the other entry from SAProuter dropdown list.

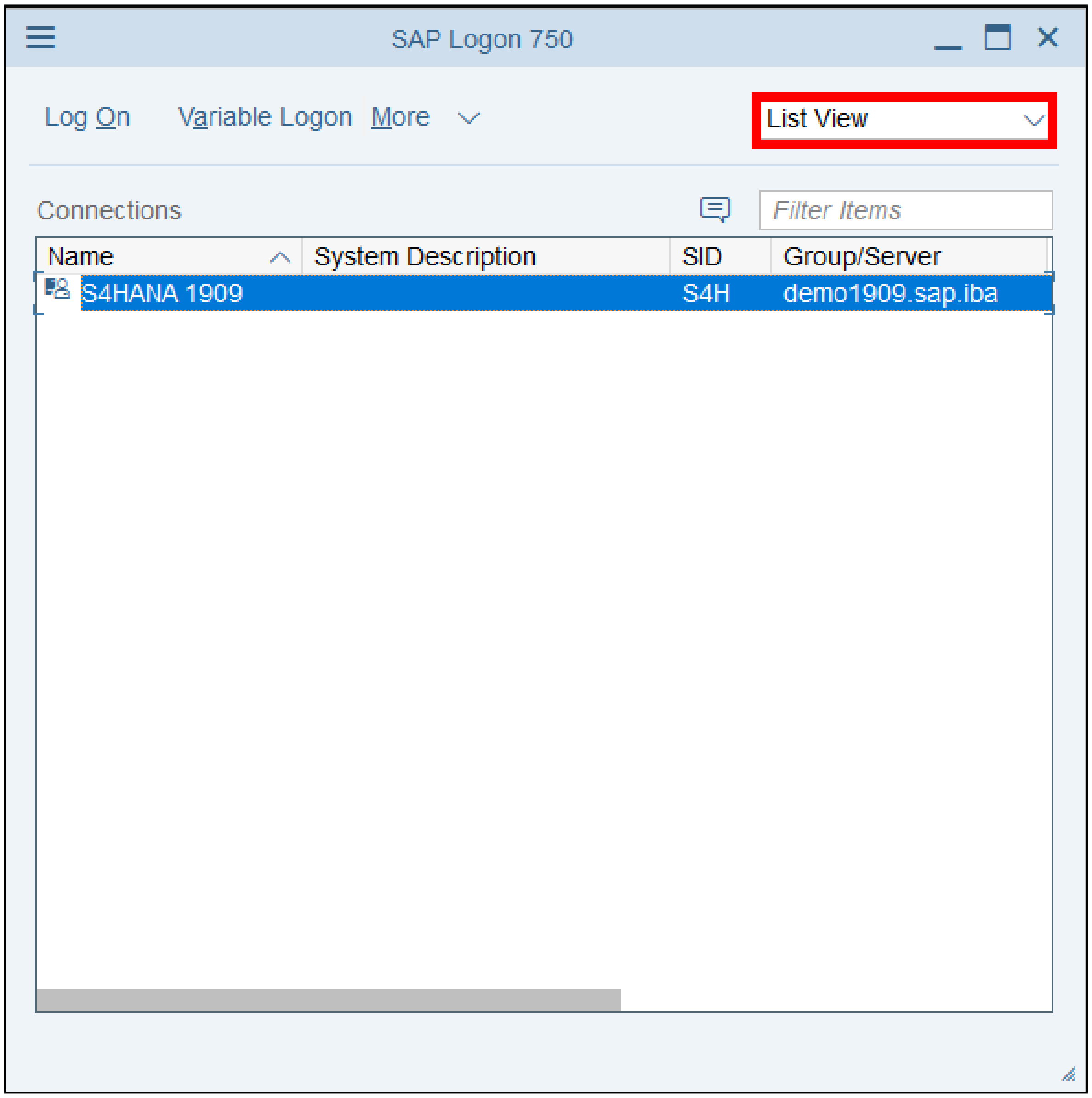
SAProuter:

Click Next and specify Description, Application Server, Instance Number and System ID:



Description:	S4HANA 1909
Application Server:	demo1909.sap.iba
Instance Number:	00
System ID:	S4H
SAProuter String:	

Finally switch to List View. This will prevent the connection entry from being hided during occasional tabs switching.



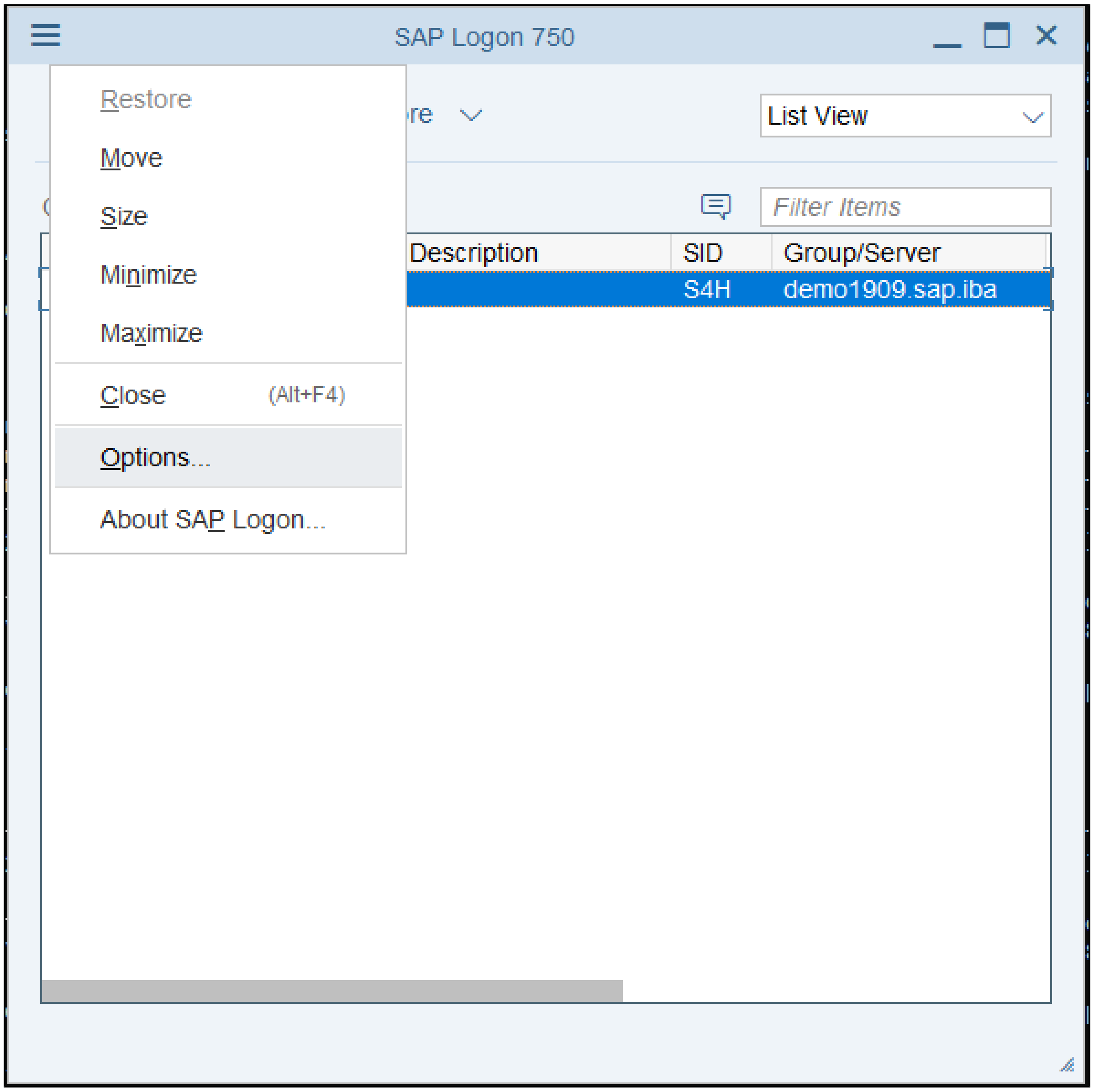
Name	System Description	SID	Group/Server
S4HANA 1909		S4H	demo1909.sap.iba

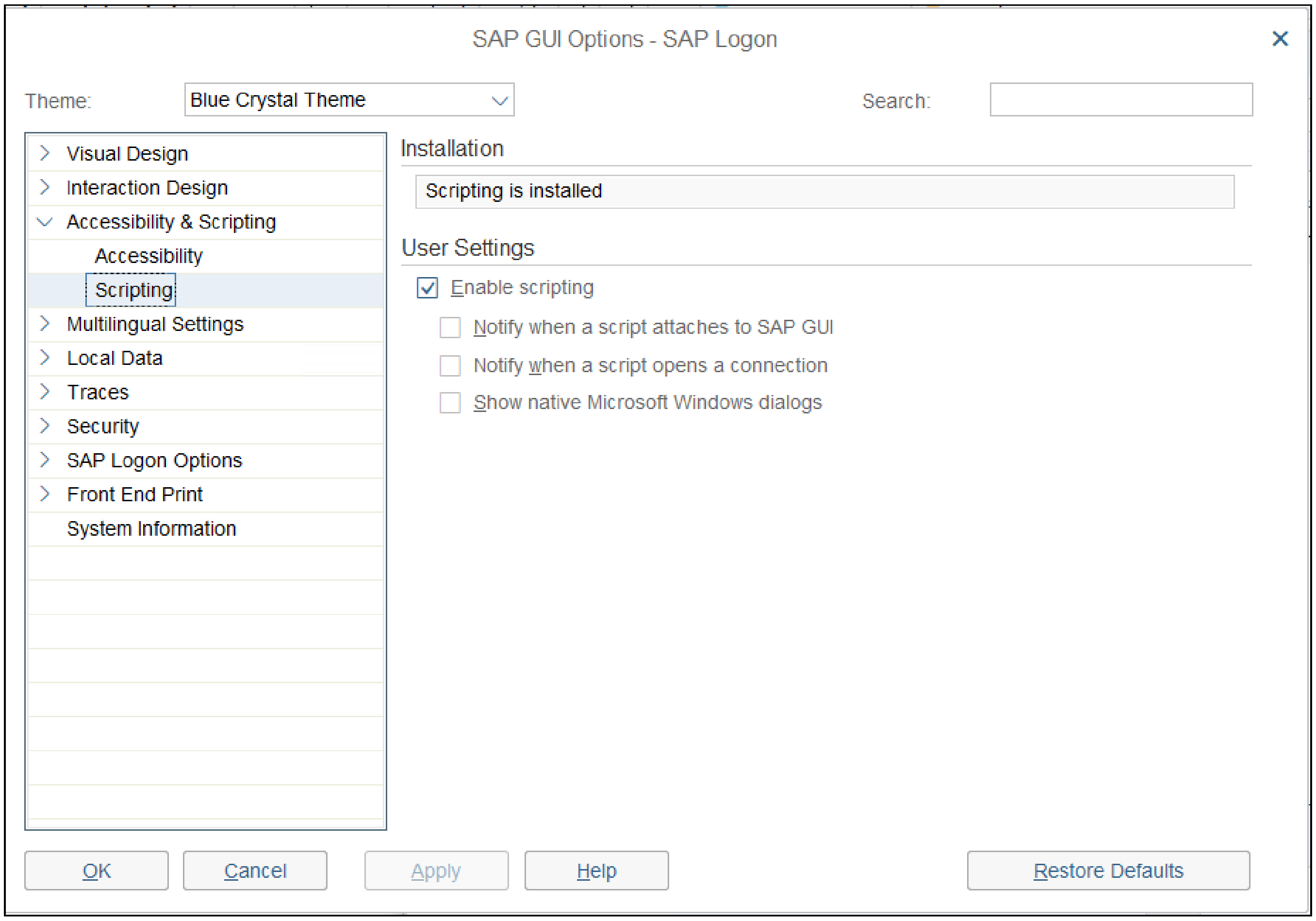
Enable scripting

It is recommended to download Tracker from https://tracker.stschnell.de/

One must be able to run Tracker against SAP Frontend and see its object tree after completing this step

Enable scripting on client side, switch off the notifications:




SAP GUI Options - SAP Logon

Theme: Blue Crystal Theme

Search:

- Visual Design
- Interaction Design
- Accessibility & Scripting
  - Accessibility
  - Scripting
- Multilingual Settings
- Local Data
- Traces
- Security
- SAP Logon Options
- Front End Print
- System Information

Installation

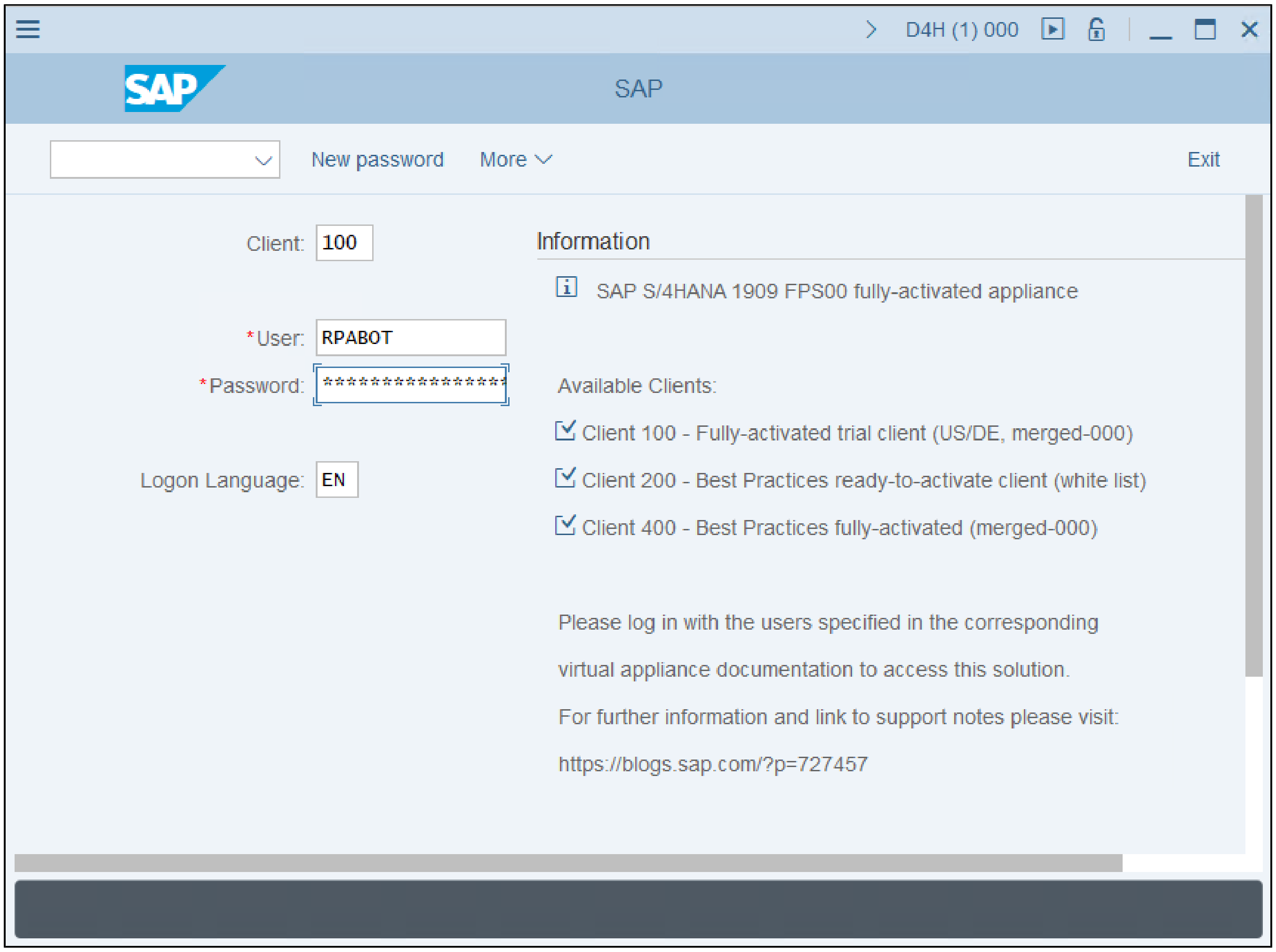
Scripting is installed

User Settings

- [✓] Enable scripting
- [ ] Notify when a script attaches to SAP GUI
- [ ] Notify when a script opens a connection
- [ ] Show native Microsoft Windows dialogs

Enable scripting on server:

Log into created connection using your credentials



Client: 100

*User: RPABOT

*Password: ****************

Logon Language: EN

Information

SAP S/4HANA 1909 FPS00 fully-activated appliance

Available Clients:

Client 100 - Fully-activated trial client (US/DE, merged-000)

Client 200 - Best Practices ready-to-activate client (white list)

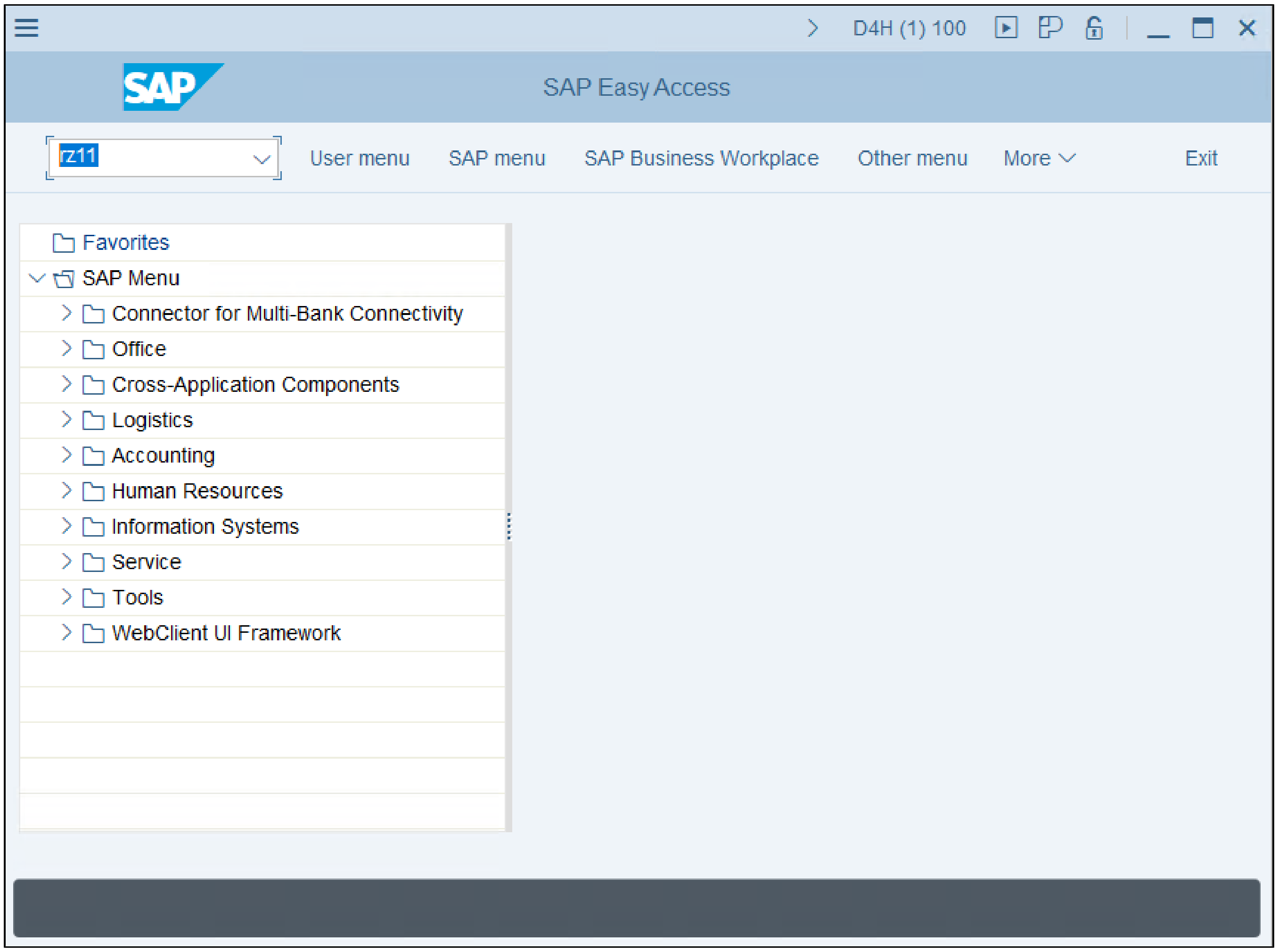
Client 400 - Best Practices fully-activated (merged-000)

Please log in with the users specified in the corresponding virtual appliance documentation to access this solution.

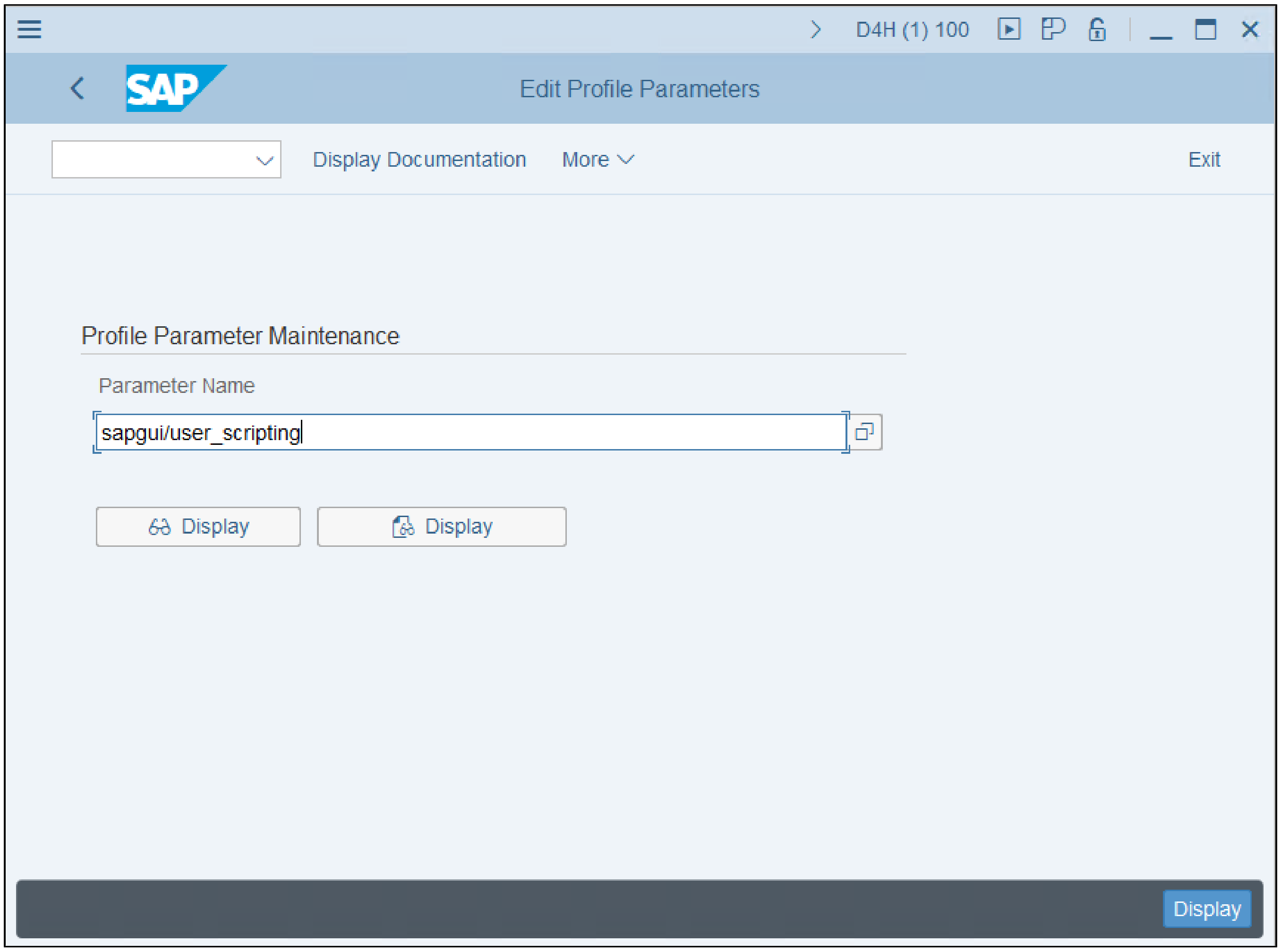
For further information and link to support notes please visit:

https://blogs.sap.com/?p=727457

Run rz11 transaction code



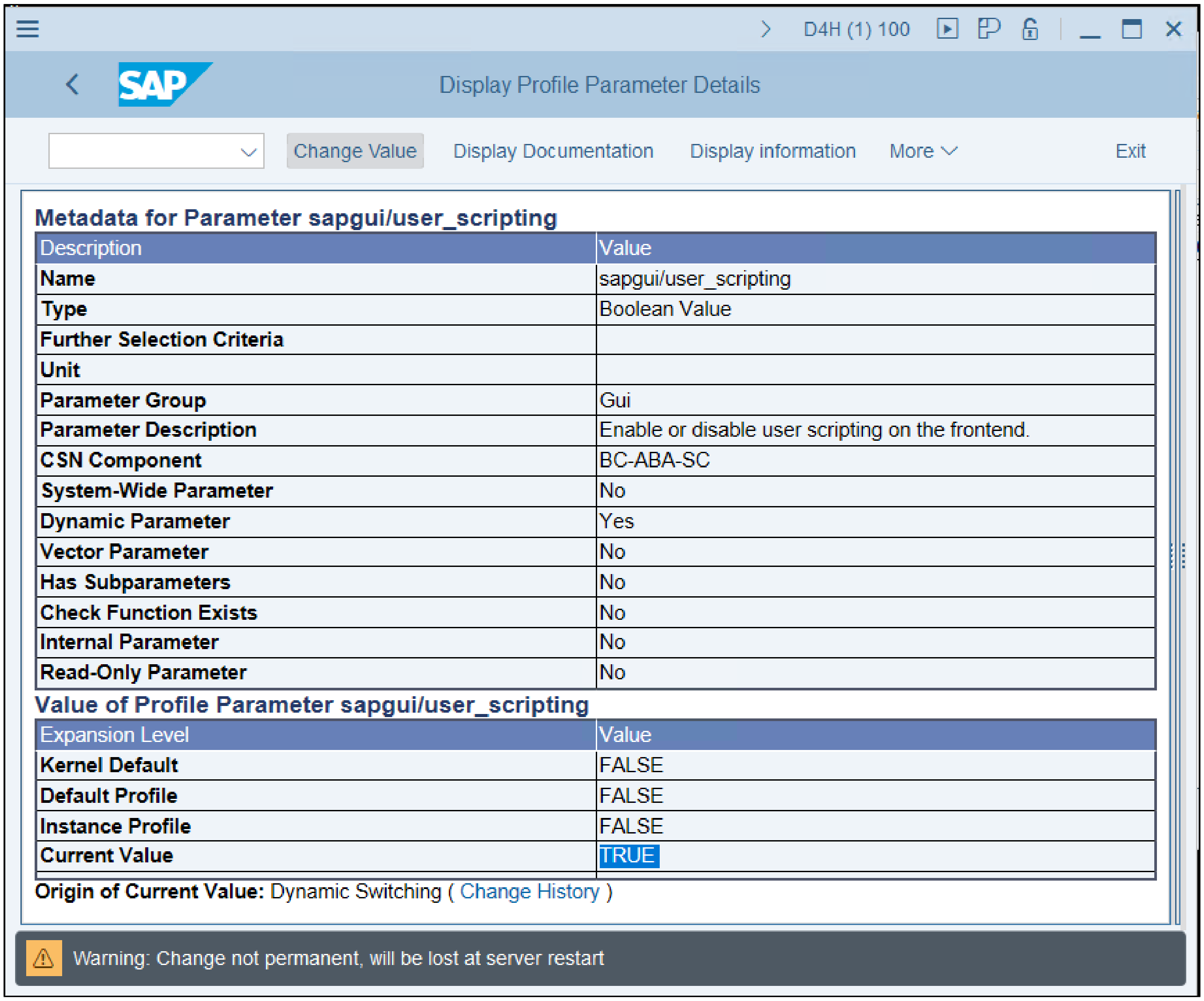
Type in `sapgui/user_scripting` parameter name and press Enter



Profile Parameter Maintenance

Parameter Name: sapgui/user_scripting

Make sure current value is set to TRUE, otherwise change it using upper menu



Metadata for Parameter sapgui/user_scripting

Description	Value
Name	sapgui/user_scripting
Type	Boolean Value
Further Selection Criteria	
Unit	
Parameter Group	Gui
Parameter Description	Enable or disable user scripting on the frontend.
CSN Component	BC-ABA-SC
System-Wide Parameter	No
Dynamic Parameter	Yes
Vector Parameter	No
Has Subparameters	No
Check Function Exists	No
Internal Parameter	No
Read-Only Parameter	No

Value of Profile Parameter sapgui/user_scripting

Expansion Level	Value
Kernel Default	FALSE
Default Profile	FALSE
Instance Profile	FALSE
Current Value	TRUE

Origin of Current Value: Dynamic Switching (Change History)

Warning: Change not permanent, will be lost at server restart

Demo overview

SAP short example consists of the following steps:

1. start SAP session with a certain transaction using sapshcut.exe
2. perfrom simple actions such as clicks and text input
3. logoff and terminate session

apm_run.properties file stores settings required to start application.

apm_run.properties

```
app.sap.path=C:/Program Files (x86)/SAP/FrontEnd/SAPgui/sapshcut

# system ID from connection properties
app.sap.sid=S4H

# here 100 is fully activated trial client
app.sap.client=100
```

where:

app.sap.sid -

app.sap.client - client

vault.properties stores SAP user credentials encoded as base64

Read more about secret storage in (v. 2.2) Vault Service

vault.properties

```
sap.user=eyAidXNlciI6ICJSUEFCT1QiLCAicGFzc3dvcmQiOiAiUGFzc3dvcmQxIiB9
```

SapFrontendApplication class provides two methods for SAP window management:

- `open(String... args)` - initiates SAP Frontend session by starting sapshcut executable with aforementioned settings
- `close()` - terminates SAP process with all related windows

Here transaction MB52 opens immediately on start which corresponds to list of warehouse stocks

SapFrontendApplication.java

```
package eu.ibagroup.easyrpa.sap;

import eu.ibagroup.easyrpa.engine.rpa.Application;
import eu.ibagroup.easyrpa.engine.rpa.driver.SapDriver;
import eu.ibagroup.easyrpa.engine.rpa.element.SapElement;
import eu.ibagroup.easyrpa.sap.page.ReportSetupPage;
import lombok.extern.slf4j.Slf4j;

import java.io.IOException;

@Slf4j
public class SapFrontendApplication extends Application<SapDriver, SapElement> {
        public SapFrontendApplication(SapDriver driver) {
                super(driver);
        }

        @Override
        public ReportSetupPage open(String... args) {
                String sapPath = args[0];
                String user = args[1];
                String password = args[2];
                String system = args[3];
                String client = args[4];
                String tCode = "MB52";

                String command = String.format("\"%s\" -user=%s -pw=%s -language=EN -system=%s -client=%s -
command=%s -trace=0", sapPath, user, password, system, client, tCode);

                try {
                        Runtime.getRuntime().exec(command);

                        try {
                                Thread.sleep(6000);
                        } catch (InterruptedException e) {
                                log.error(e.getMessage(), e);
                        }
                        getDriver().get(null);
                        return createPage(ReportSetupPage.class);
                } catch (IOException e) {
                        log.error(e.getMessage(), e);
                }
                return null;
        }

        @Override
        public void close() {
                try {
                        Runtime.getRuntime().exec("TASKKILL /F /IM saplogon.exe /T");
                } catch (IOException e) {
                        log.error(e.getMessage(), e);
                }
        }
}
```

Next follow two SAP page objects

ReportsSetupPage contains handles for setting up warehouse stock report prior to its execution

Here `materialField` and `plantField` are respectively material code and plant code placeholders

window corresponds to entire SAP window which is handy to call hotkeys against, such as F8 (execute report)

Read more about SAP xpath selectors: How to Locate Interface Elements#ByXPath.1

ReportSetupPage.java

```
package eu.ibagroup.easyrpa.sap.page;

import eu.ibagroup.easyrpa.engine.rpa.element.SapElement;
import eu.ibagroup.easyrpa.engine.rpa.page.SapPage;
import lombok.extern.slf4j.Slf4j;
import org.openqa.selenium.Keys;
import org.openqa.selenium.support.FindBy;

@Slf4j
public class ReportSetupPage extends SapPage {

        @FindBy(id = "/app/con[0]/ses[0]/wnd[0]/usr/ctxtMATNR-LOW")
        private SapElement materialField;

        @FindBy(id = "/app/con[0]/ses[0]/wnd[0]/usr/ctxtWERKS-LOW")
        private SapElement plantField;

        @FindBy(id = "/app/con[0]/ses[0]/wnd[0]")
        private SapElement window;

        public void setupReport() {
                materialField.setText("TG11");
                plantField.setText("1010");
        }

        public ReportResultPage generateReport() {
                window.sendKeys(Keys.F8);

                try {
                        Thread.sleep(3000);
                } catch (InterruptedException e) {
                        log.error(e.getMessage(), e);
                }

                return this.getApplication().createPage(ReportResultPage.class);
        }
}
```

ReportResultPage represents ALV (ABAP List Viewer) which is basically a table layout containing report result

We only implement `close()` method since we're not going to perform any further actions on this page

ReportResultPage.java

```
package eu.ibagroup.easyrpa.sap.page;

import eu.ibagroup.easyrpa.engine.rpa.page.SapPage;

public class ReportResultPage extends SapPage {

        public void close() {
                getDriver().close();
        }
}
```

Finally let's bring together application initialization, pages creation and action calls in **WarehouseStock** task

```
package eu.ibagroup.easyrpa.sap.task;

import eu.ibagroup.easyrpa.engine.annotation.AfterInit;
import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.annotation.Driver;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.engine.model.SecretCredentials;
import eu.ibagroup.easyrpa.engine.rpa.driver.DriverParams;
import eu.ibagroup.easyrpa.engine.rpa.driver.SapDriver;
import eu.ibagroup.easyrpa.sap.SapFrontendApplication;
import eu.ibagroup.easyrpa.sap.page.ReportResultPage;
import eu.ibagroup.easyrpa.sap.page.ReportSetupPage;

@ApTaskEntry(name = "Get warehouse stocks of material")
public class WarehouseStock extends ApTask {
        private SecretCredentials credentials;

        @Driver(DriverParams.Type.SAP)
        private SapDriver sapDriver;

        private String sapPath;

        private String sapSystemID;

        private String sapClient;

        @AfterInit
        public void init() {
                credentials = getVaultService().getSecret("sap.user", SecretCredentials.class);
                sapPath = getConfigurationService().get("app.sap.path", "C:/Program Files (x86)/SAP/FrontEnd
/SAPgui/sapshcut");
                sapSystemID = getConfigurationService().get("app.sap.sid", "S4H");
                sapClient = getConfigurationService().get("app.sap.client", "100");
        }

        @Override
        public void execute() {
                SapFrontendApplication sapLogonApplication = new SapFrontendApplication(sapDriver);
                ReportSetupPage setupPage = sapLogonApplication.open(sapPath, credentials.getUser(),
credentials.getPassword(), sapSystemID, sapClient);
                setupPage.setupReport();
                ReportResultPage resultPage = setupPage.generateReport();
                resultPage.close();
                sapDriver.quit();
                sapLogonApplication.close();
        }
}
```

(v. 2.2) Screen Based Automation

This is the example of automating Windows 10 calclator using screen based automation.

Steps include:

1. Open calculator using start menu
2. Maximize calculator window and switch to scientific mode
3. Calculate logarithm
4. Extract result from screen using OCR
5. Close calculator window

Application class returns `WindowsPage` object which provides access to Windows 10 desktop.

WindowsScreenApplication .java

```
package eu.ibagroup.easyrpa.calculator;

import eu.ibagroup.easyrpa.calculator.page.WindowsPage;
import eu.ibagroup.easyrpa.engine.rpa.Application;
import eu.ibagroup.easyrpa.engine.rpa.driver.ScreenDriver;
import eu.ibagroup.easyrpa.engine.rpa.element.ImageElement;

public class WindowsScreenApplication extends Application<ScreenDriver, ImageElement> {

        public WindowsScreenApplication(ScreenDriver driver) {
                super(driver);
        }

        @Override
        public WindowsPage open(String... args) {
                return createPage(WindowsPage.class);
        }
}
```

`WindowsPage` only contains a single public method `openCalcApp()` which opens calculator main window represented as `CalculatorMainPage`

WindowsPage.java

```
package eu.ibagroup.easyrpa.calculator.page;

import eu.ibagroup.easyrpa.engine.rpa.element.ImageElement;
import eu.ibagroup.easyrpa.engine.rpa.page.ScreenPage;
import eu.ibagroup.easyrpa.engine.rpa.po.annotation.Image;
import eu.ibagroup.easyrpa.engine.rpa.po.annotation.WithTimeout;
import lombok.extern.slf4j.Slf4j;
import org.sikuli.script.Key;

@Slf4j
public class WindowsPage extends ScreenPage {

        private static final long ANIMATION_TIMEOUT = 1000;

        @Image(url = "images/windows-start-1.png")
        @WithTimeout(time = 3)
        private ImageElement start;

        public CalculatorMainPage openCalcApp() {
                launchFromStart("calculator");
                sleep(ANIMATION_TIMEOUT);

                return createPage(CalculatorMainPage.class);
        }
```

```
	private void launchFromStart(String appName) {
		start.click();
		sleep(ANIMATION_TIMEOUT);
		getDriver().sendKeys(appName);
		sleep(ANIMATION_TIMEOUT);
		getDriver().sendKeys(Key.ENTER);
	}

	private static void sleep(long millis) {
		try {
			Thread.sleep(millis);
		} catch (InterruptedException e) {
			log.error(e.getMessage(), e);
		}
	}
}
```

`CalculatorMainPage` can potentially provide handles for all the buttons and menus rendered on calculator window, but we restricted those to demo purposes.

CalculatorMainPage.java

```
package eu.ibagroup.easyrpa.calculator.page;

import eu.ibagroup.easyrpa.engine.rpa.element.ImageElement;
import eu.ibagroup.easyrpa.engine.rpa.page.ScreenPage;
import eu.ibagroup.easyrpa.engine.rpa.po.annotation.Image;
import eu.ibagroup.easyrpa.engine.rpa.po.annotation.WithTimeout;
import lombok.extern.slf4j.Slf4j;
import org.sikuli.script.Key;
import org.sikuli.script.Region;

@Slf4j
public class CalculatorMainPage extends ScreenPage {

	private static final double CALC_DISPLAY_HEIGHT_RATIO = 0.14;

	private static final double CALC_DISPLAY_WIDTH_RATIO = 0.85;

	private static final double CALC_DISPLAY_Y_RATIO = 0.12;

	@Image(url = "images/btn-navigation.png")
	@WithTimeout(time = 3)
	private ImageElement navigation;

	@Image(url = "images/btn-four.png")
	@WithTimeout(time = 3)
	private ImageElement fourBtn;

	@Image(url = "images/btn-six.png")
	@WithTimeout(time = 3)
	private ImageElement sixBtn;

	@Image(url = "images/btn-eight.png")
	@WithTimeout(time = 3)
	private ImageElement eightBtn;

	@Image(url = "images/btn-plus.png")
	@WithTimeout(time = 3)
	private ImageElement plusBtn;

	@Image(url = "images/btn-equals")
	@WithTimeout(time = 3)
	private ImageElement equalsBtn;

	public void four() {
		fourBtn.click();
	}
	public void six() {
		sixBtn.click();
	}

	public void eight() {
		eightBtn.click();
	}

	public void plus() {
		plusBtn.click();
	}

	public void equals() {
		equalsBtn.click();
	}

	public String getDisplayText() {
		int h = (int) (CALC_DISPLAY_HEIGHT_RATIO * getDriver().getScreen().h);
		int w = (int) (CALC_DISPLAY_WIDTH_RATIO * getDriver().getScreen().w);
		int y = (int) (CALC_DISPLAY_Y_RATIO * getDriver().getScreen().h);
		int x = 5;

		log.debug("Reading region: x={} y={} w={} h={}", x, y, w, h);

		return getRegionText(x, y, w, h);
	}

	private String getRegionText(int x, int y, int w, int h) {
		Region regResult = getDriver().getScreen().newRegion(x, y, w, h);
		regResult.highlight(2);

		return regResult.text();
	}

	public void exit() {
		getDriver().getScreen().keyDown(Key.ALT);
		getDriver().getScreen().keyDown(Key.F4);
		getDriver().getScreen().keyUp();
	}

	public void maximizeWindow() {
		getDriver().getScreen().keyDown(Key.ALT);
		getDriver().getScreen().keyDown(Key.SPACE);
		getDriver().getScreen().keyDown("x");
		getDriver().getScreen().keyUp();
	}
}
```

`getDisplayText()` is worth a specific mention since it employs OCR to read a certain region on screen. This region is basically a rectangular defined by upper left vertex (x, y), height (h) and width (w).

(v. 2.2) Taking Screenshots

There're helpful methods to take screenshots in **any** driver:

-
  ```
  byte[] bytes = driver.getScreenshotAsBytes();
  ```

-
  ```
  File file = driver.getScreenshotAsFile();
  ```

-
  ```
  String base64String = driver.getScreenshotAsBase64();
  ```

Example of usage:

TakeScreenshot.java

```
@ApTaskEntry(name = "Take screenshot")
public class TakeScreenshot extends ApTask {

        @Driver(DriverParams.Type.DESKTOP)
        private DesktopDriver desktopDriver;

        @Override
        public void execute() throws IOException {
                byte[] bytes = desktopDriver.getScreenshotAsBytes();
                FileUtils.writeByteArrayToFile(new File("C:\\work\\screenshot.png"), bytes);
        }
}
```

(v. 2.2) Upload file

This example illustrates how to automate file upload in a web application, as well as using two drivers in one task.

Application class opens web browser and returns MainPage object.

ViljamisApplication.java

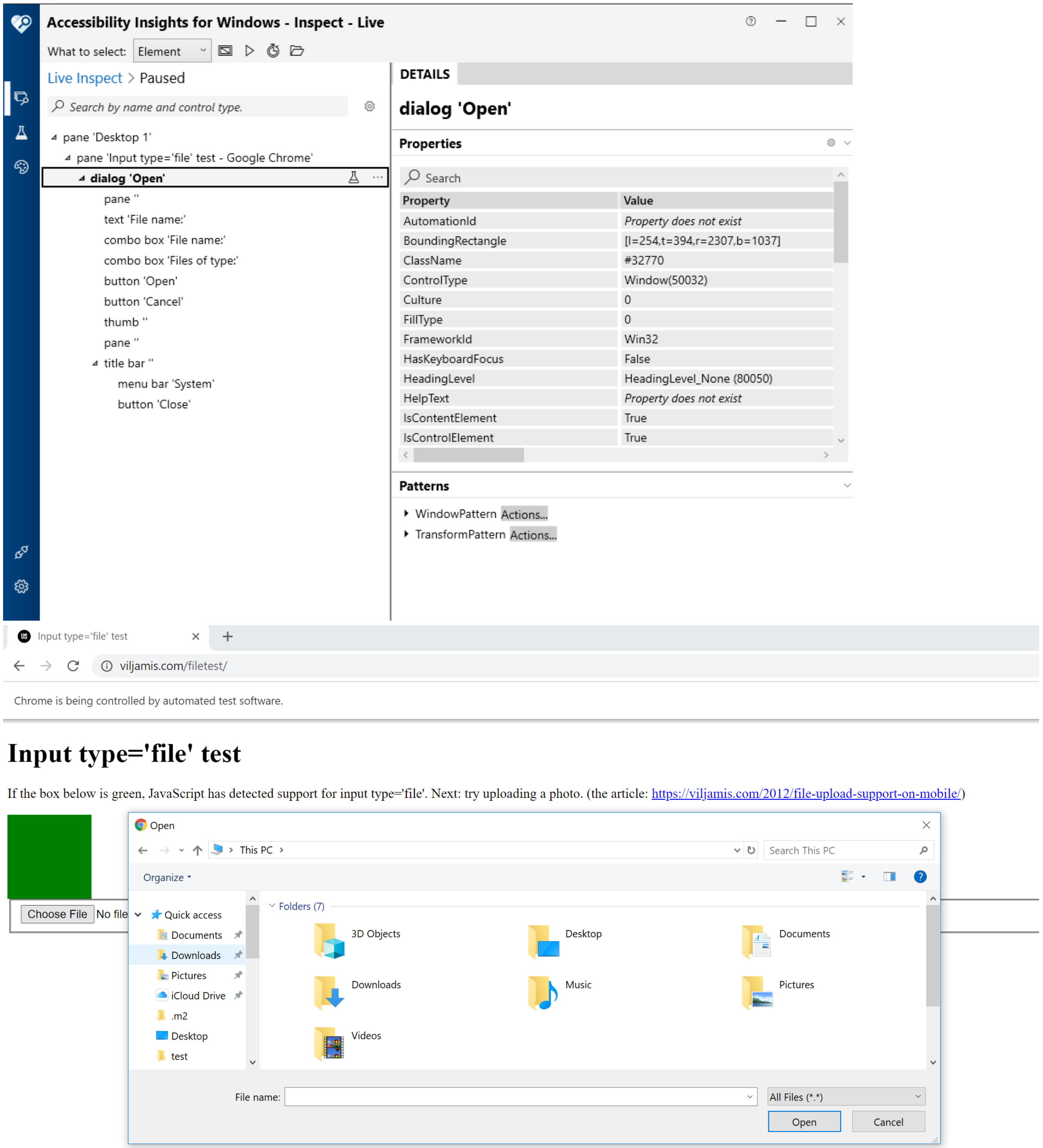
```
package eu.ibagroup.easyrpa.fileupload;

import eu.ibagroup.easyrpa.engine.rpa.Application;
import eu.ibagroup.easyrpa.engine.rpa.driver.BrowserDriver;
import eu.ibagroup.easyrpa.engine.rpa.element.UiElement;
import eu.ibagroup.easyrpa.fileupload.page.MainPage;

public class ViljamisApplication extends Application<BrowserDriver, UiElement> {

        public ViljamisApplication(BrowserDriver driver) {
                super(driver);
        }

        @Override
        public MainPage open(String... args) {
                String gmailUrl = args[0];
                getDriver().get(gmailUrl);
                return createPage(MainPage.class);
        }
}
```

The MainPage class contains method openFileWindow(DesktopDriver). This method opens an "Open" window, so it should return an OpenFilePage class object. However, OpenFilePage refers to another application, which is responsible for its creation. Therefore, the method receives as a parameter the driver to work with the application it opens, then it performs actions to open an "Open" window, creates the necessary application and returns the necessary page by calling the corresponding method from the application.

MainPage.java

```
package eu.ibagroup.easyrpa.fileupload.page;

import eu.ibagroup.easyrpa.engine.rpa.driver.DesktopDriver;
import eu.ibagroup.easyrpa.engine.rpa.page.WebPage;
import eu.ibagroup.easyrpa.engine.rpa.po.annotation.Wait;
import eu.ibagroup.easyrpa.fileupload.OpenFileApplication;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.FindBy;

public class MainPage extends WebPage {

        @FindBy(xpath = "//input[@name='image']")
        @Wait(waitFunc = Wait.WaitFunc.VISIBLE)
        private WebElement chooseFileInput;

        @FindBy(xpath = "//input[@type='submit']")
        @Wait(waitFunc = Wait.WaitFunc.VISIBLE)
        private WebElement uploadInput;

        public OpenFilePage openFileWindow(DesktopDriver desktopDriver) {
                Actions builder = new Actions(getDriver());
                builder.moveToElement(chooseFileInput).click().perform();

                OpenFileApplication openFileApplication = new OpenFileApplication(desktopDriver);
                return openFileApplication.open();
        }
        public void upload() {
                uploadInput.click();
        }
}
```

Structurally, the file opening window is not a separate window, but an element of the browser window.


Accessibility Insights for Windows - Inspect - Live

What to select: Element

Live Inspect > Paused

```
pane 'Desktop 1'
  pane 'Input type='file' test - Google Chrome'
    dialog 'Open'
      pane ''
      text 'File name:'
      combo box 'File name:'
      combo box 'Files of type:'
      button 'Open'
      button 'Cancel'
      thumb ''
      pane ''
      title bar ''
        menu bar 'System'
        button 'Close'
```

DETAILS

dialog 'Open'

Properties

Property	Value
AutomationId	*Property does not exist*
BoundingRectangle	[l=254,t=394,r=2307,b=1037]
ClassName	#32770
ControlType	Window(50032)
Culture	0
FillType	0
FrameworkId	Win32
HasKeyboardFocus	False
HeadingLevel	HeadingLevel_None (80050)
HelpText	*Property does not exist*
IsContentElement	True
IsControlElement	True

Patterns

- WindowPattern Actions...
- TransformPattern Actions...

Input type='file' test

viljamis.com/filetest/

Chrome is being controlled by automated test software.

Input type='file' test

If the box below is green, JavaScript has detected support for input type='file'. Next: try uploading a photo. (the article: https://viljamis.com/2012/file-upload-support-on-mobile/)

Structurally, the file opening window is not a separate window, but an element of the browser window. Therefore, to access the "Open" window elements, we need to switch the DesktopDriver to the main browser window.

The OpenFilePage class stores a string with the window name and switches in the init method. When creating an object, this method is executed and switches the driver to a browser window, giving access to its elements.

The ChooseFile method enters the path to the file and clicks the "Open" button.

OpenFilePage.java

```
package eu.ibagroup.easyrpa.fileupload.page;

import eu.ibagroup.easyrpa.engine.annotation.AfterInit;
import eu.ibagroup.easyrpa.engine.rpa.element.DesktopElement;
import eu.ibagroup.easyrpa.engine.rpa.page.DesktopPage;
import eu.ibagroup.easyrpa.engine.rpa.po.annotation.WithTimeout;
import org.openqa.selenium.support.FindBy;

public class OpenFilePage extends DesktopPage {

        @FindBy(className = "Edit", name = "File name:")
        @WithTimeout(time = 3)
        private DesktopElement fileNameInput;

        @FindBy(className = "Button", name = "Open")
        @WithTimeout(time = 3)
        private DesktopElement openBtn;

        @AfterInit
        public void init() {
                String title = "Input type='file' test - Google Chrome";
                getDriver().waitAndSwitchToWindow(title, 5); // switch to the root Chrome window first
                getDriver().waitAndSwitchToWindow("Open", 5);
        }

        public void chooseFile(String fullFilePath) {
                fileNameInput.sendKeys(fullFilePath);
                openBtn.click();
        }
}
```

`UploadFile- task-class containing the main algorithm of the task.`

CreateNote.java

```
package eu.ibagroup.easyrpa.fileupload.task;

import eu.ibagroup.easyrpa.engine.annotation.AfterInit;
import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.annotation.Driver;
import eu.ibagroup.easyrpa.engine.annotation.DriverParameter;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.engine.rpa.driver.BrowserDriver;
import eu.ibagroup.easyrpa.engine.rpa.driver.DesktopDriver;
import eu.ibagroup.easyrpa.engine.rpa.driver.DriverParams;
import eu.ibagroup.easyrpa.fileupload.ViljamisApplication;
import eu.ibagroup.easyrpa.fileupload.page.MainPage;
import eu.ibagroup.easyrpa.fileupload.page.OpenFilePage;

@ApTaskEntry(name = "Get product list from InvoicePlane")
public class UploadFile extends ApTask {

        @Driver(value = DriverParams.Type.BROWSER, param = {
        @DriverParameter(key = DriverParams.Browser.SELENIUM_NODE_CAPABILITIES, initializer = BrowserDriver.
DefaultChromeOptions.class) })
        private BrowserDriver browserDriver;
        @Driver(DriverParams.Type.DESKTOP)
        private DesktopDriver desktopDriver;

        private String viljamisUrl;

        private String filePath;

        private String fileName;

        @AfterInit
        public void init() {
                viljamisUrl = getConfigurationService().get("viljamis.app.url", "https://viljamis.com/filetest
/");
                filePath = getConfigurationService().get("viljamis.file.path", System.getProperty("user.home"));
                fileName = getConfigurationService().get("viljamis.file.name", "AboutNotepad.txt");
        }

        @Override
        public void execute() {
                ViljamisApplication viljamisApplication = new ViljamisApplication(browserDriver);
                MainPage mainPage = viljamisApplication.open(viljamisUrl);

                OpenFilePage openFilePage = mainPage.openFileWindow(desktopDriver);

                openFilePage.chooseFile(filePath + "\\" + fileName);
                mainPage.upload();
        }
}
```

Automation process and runner classes.

UploadFileAp .java

```
package eu.ibagroup.easyrpa.fileupload;

import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModule;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import eu.ibagroup.easyrpa.fileupload.task.UploadFile;

@ApModuleEntry(name = "File Upload Automation Demo")
public class UploadFileAp extends ApModule {

        public TaskOutput run() throws Exception {
                return execute(getInput(), UploadFile.class).get();
        }
}
```

LocalRunner.java

```
package eu.ibagroup.easyrpa.fileupload;

import eu.ibagroup.easyrpa.engine.boot.ApModuleRunner;

public class LocalRunner {

        public static void main(String[] args) {
                ApModuleRunner.localLaunch(UploadFileAp.class);
        }
}
```

(v. 2.2) Oracle Forms

Introduction

This sample demonstrates the automation of Oracle Forms application

Prerequisites

to run the Automation process (AP), the node should have Capabilities specified in the AP Details tab. Capabilities for Oracle Forms: AP_RUN, JAVA

repo id: eu.ibagroup:kb-short-samples:jar:full:<easyRPA version>

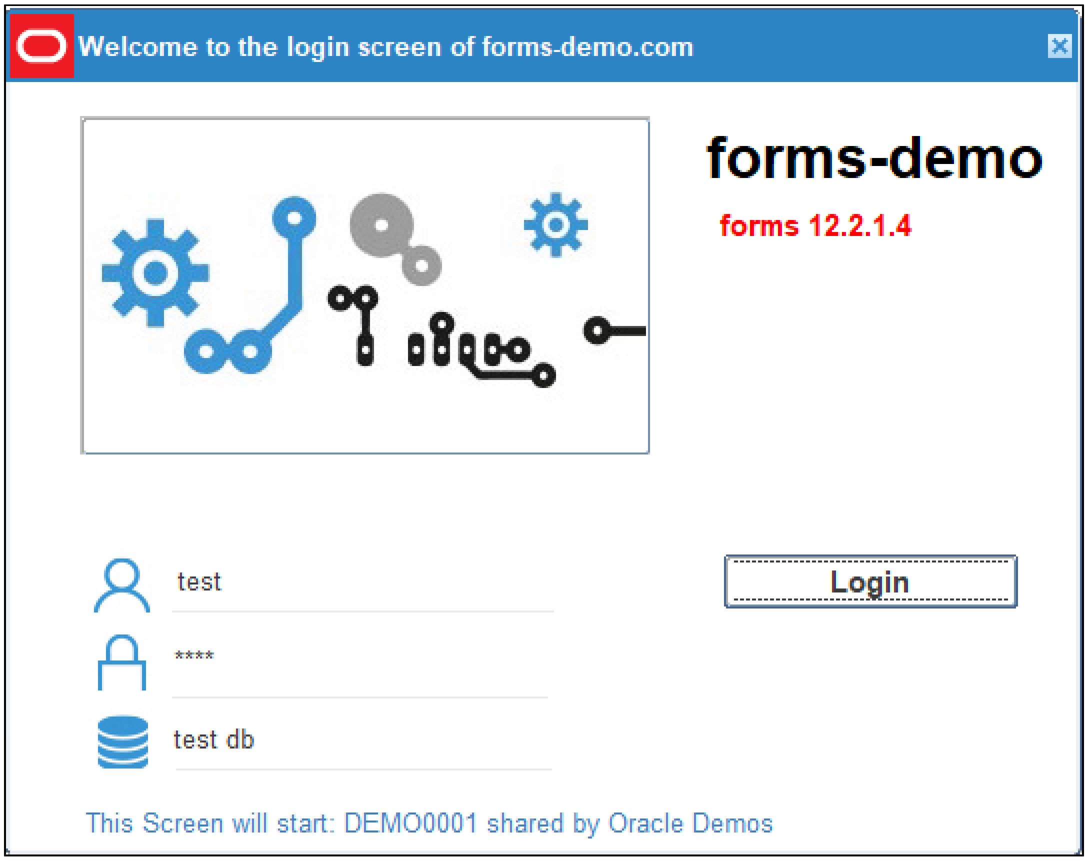
module class: eu.ibagroup.easyrpa.oracleforms.OracleFormsAp

configuration params:

key	default value	required	description
app.path	app/start-oracle-forms-app.bat	no	full path to Oracle Forms executable start-oracle-forms-app.bat

Included Steps

The bot opens link to Oracle Forms demo in Internet Explorer and after some time the login window appears


Welcome to the login screen of forms-demo.com

forms-demo

forms 12.2.1.4

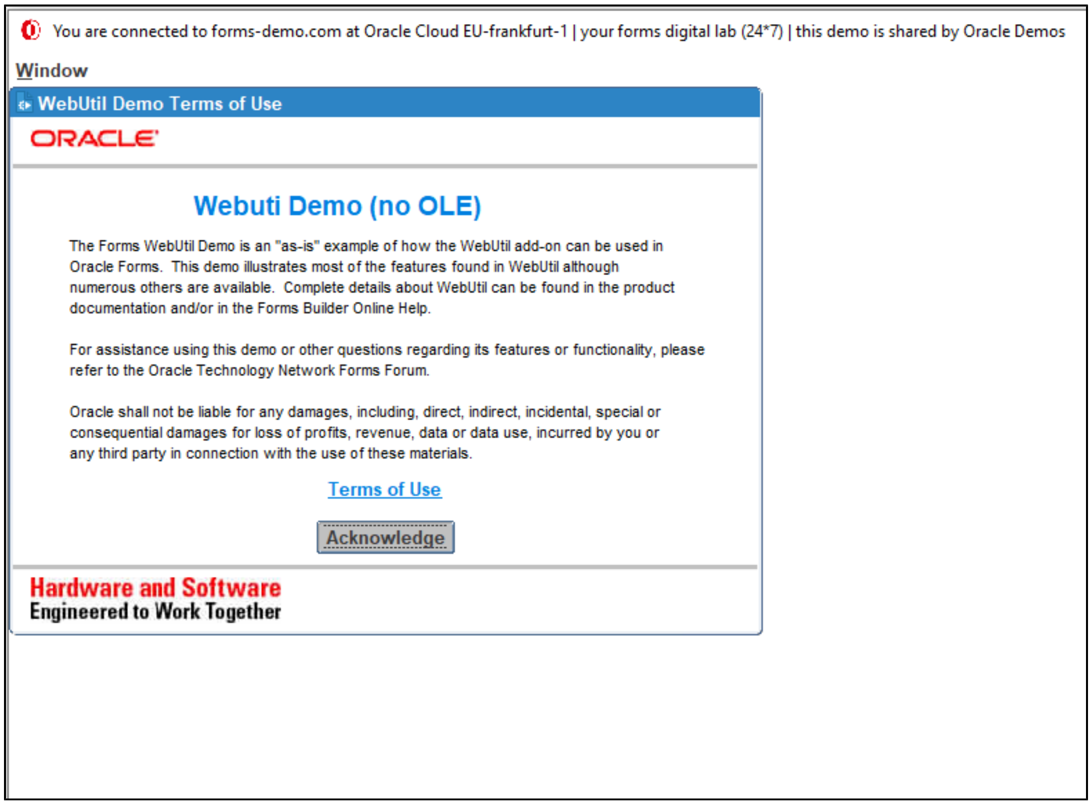
test

****

test db

This Screen will start: DEMO0001 shared by Oracle Demos

The bot clicks *Login* button leaving the rest of fields as is


You are connected to forms-demo.com at Oracle Cloud EU-frankfurt-1 | your forms digital lab (24*7) | this demo is shared by Oracle Demos

Window

WebUtil Demo Terms of Use

ORACLE

Webuti Demo (no OLE)

The Forms WebUtil Demo is an "as-is" example of how the WebUtil add-on can be used in Oracle Forms. This demo illustrates most of the features found in WebUtil although numerous others are available. Complete details about WebUtil can be found in the product documentation and/or in the Forms Builder Online Help.

For assistance using this demo or other questions regarding its features or functionality, please refer to the Oracle Technology Network Forms Forum.

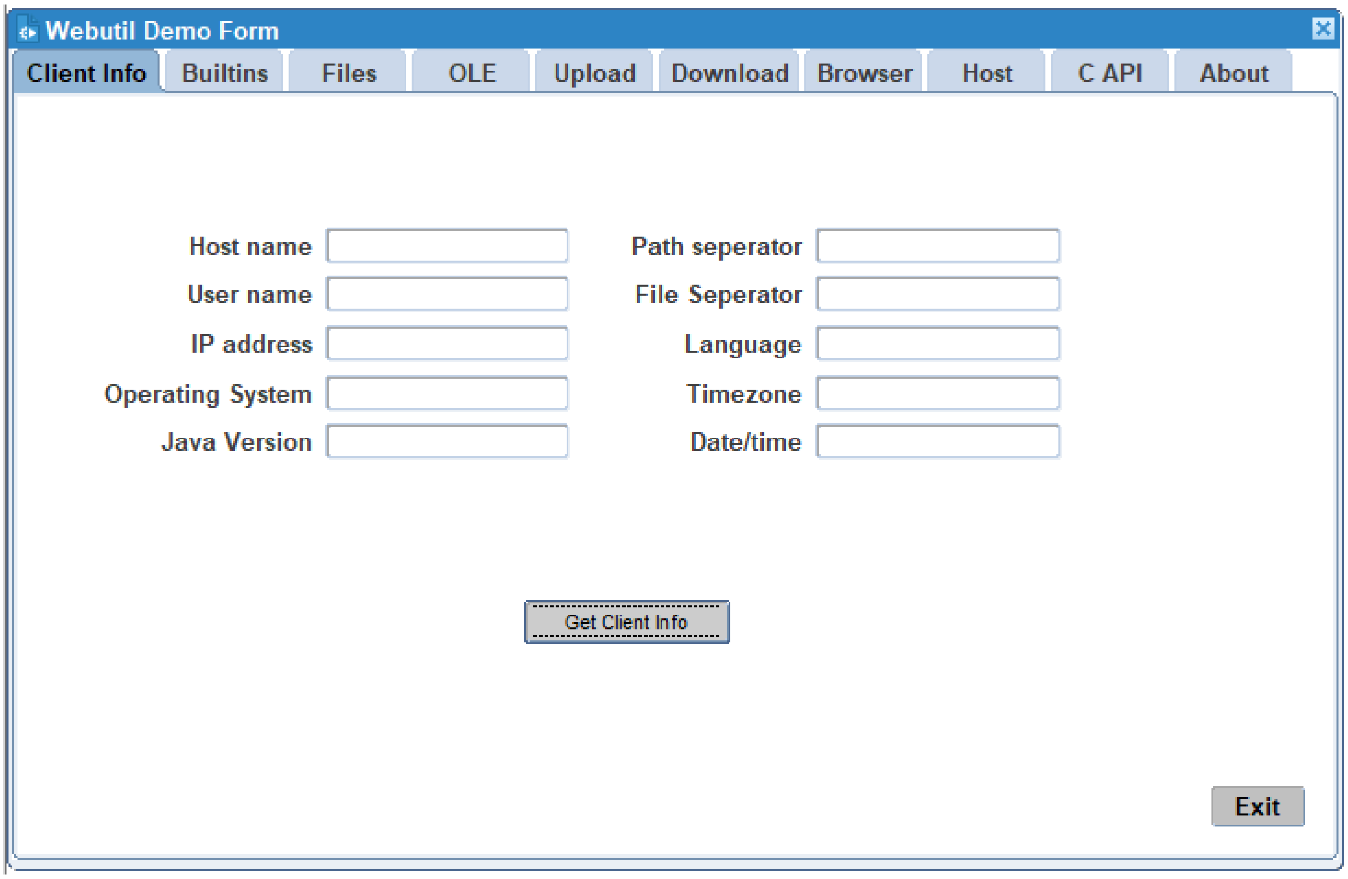
Oracle shall not be liable for any damages, including, direct, indirect, incidental, special or consequential damages for loss of profits, revenue, data or data use, incurred by you or any third party in connection with the use of these materials.

Terms of Use

Acknowledge

Hardware and Software
Engineered to Work Together

Next on terms of use screen the bot clicks *Acknowledge*



Next on client info screen the bot clicks *Get Client Info*, reads the data from fields, prints it to log and finally clicks *Exit*

(v. 2.2) Log Management

Introduction

EasyRPA provides logging infrastructure backed by logback java library.

In this section we explain how developer can make use of the logging system components to manage the logs effectively.

Defining Logger

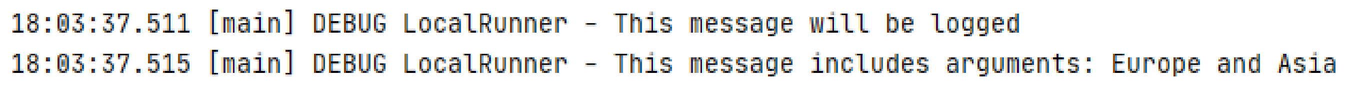
All system events publish to a single logging facade which redirects them further in accordance with configuration.

To comply with the rest of the system a developer must define Slf4j logger and start logging to it.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

class LocalRunner {
    private static final Logger log = LoggerFactory.getLogger(LocalRunner.class);

    public static void main(String[] args) {
        log.debug("This message will be logged");
        log.debug("This message includes arguments: {} and {}", "Europe", "Asia");
    }
}
```

The logger declaration can be replaced with @Slf4j annotation

When running locally the above code ouputs to console:



```
18:03:37.511 [main] DEBUG LocalRunner - This message will be logged
18:03:37.515 [main] DEBUG LocalRunner - This message includes arguments: Europe and Asia
```

Configuring Logback

Pattern

Default output pattern is configured as `%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n`

This can be translated as:

- `%d{HH:mm:ss.SSS}` current date as HH:mm:ss.SSS
- `[%thread]` thread name enclosed in square brackets
- `%-5level` log level (padded to five characters)
- `%logger{36}` logger name (36 character max)
- `%msg` logged message
- `%n` new line special character

This is exactly what the logger showed in our example.

The output pattern as well as other logger settings can be changed in `logback.xml` file. This file is loaded from resource folder when java application is started.

The typical logback.xml file looks as following:

logback.xml

```
<configuration>
    <appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>
        </encoder>
    </appender>

    <root level="DEBUG">
        <appender-ref ref="CONSOLE"/>
    </root>
</configuration>
```

There are two new features introduced in file besides already familiar pattern: appender and level.

Appender

Appender handles the event sent to logger.

CONSOLE appender (with accompanying class `ch.qos.logback.core.ConsoleAppender`) redirects our logs to standard output, the console it is.

Logback also allows other outputs such as `FileAppender` etc, you can read more about them in the official documentation.

Level

Level is what constitutes the hierarchy of events. This in turn allows to filter out the events by their severity.

To date logback provides five levels: `ERROR`, `WARN`, `INFO`, `DEBUG`, `TRACE`

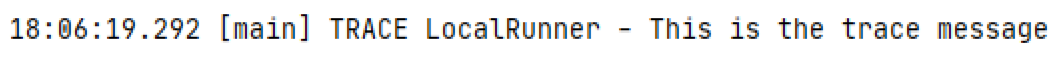
Now consider an example:

```
log.trace("This is the trace message");
```

When running this example the output shows nothing. This is because the precedence of trace level is lower than debug level configured in logback.xml

Let's change root level in logback.xml as following: `<root level="DEBUG">` and rerun the example.

This time the output log is there:



```
18:06:19.292 [main] TRACE LocalRunner - This is the trace message
```

More detailed view of logback severity levels can be found in the official documentation

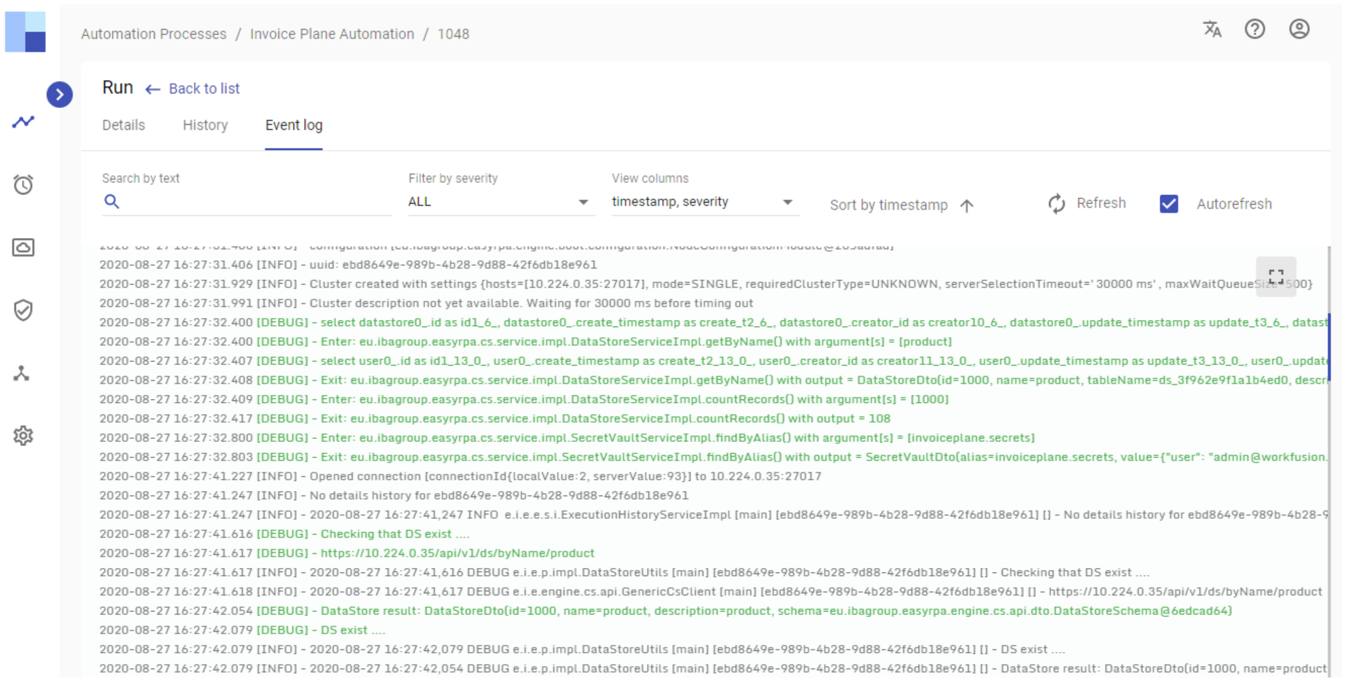
Retreiving CS Logs

Normally the developer works standalone - runs code in IDE and immediately sees the output in console, but in Control Server it's a bit different.

Contol Server stores its logs in ElasticSearch database, the same does Node Agent which runs AP.

The easiest way to check AP logs is to open Event Log:

1. go to Automation Processes list
2. click on the AP name
3. click on Run Id
4. follow to the Event log tab



```
2020-08-27 16:27:31.406 [INFO] - uuid: ebd8649e-989b-4b28-9d88-42f6db18e961
2020-08-27 16:27:31.929 [INFO] - Cluster created with settings {hosts=[10.224.0.35:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms', maxWaitQueueSize=500}
2020-08-27 16:27:31.991 [INFO] - Cluster description not yet available. Waiting for 30000 ms before timing out
2020-08-27 16:27:32.400 [DEBUG] - select datastore0_.id as id1_6_, datastore0_.create_timestamp as create_t2_6_, datastore0_.creator_id as creator10_6_, datastore0_.update_timestamp as update_t3_6_, datast
2020-08-27 16:27:32.400 [DEBUG] - Enter: eu.ibagroup.easyrpa.cs.service.impl.DataStoreServiceImpl.getByName() with argument[s] = [product]
2020-08-27 16:27:32.407 [DEBUG] - select user0_.id as id1_13_0_, user0_.create_timestamp as create_t2_13_0_, user0_.creator_id as creator11_13_0_, user0_.update_timestamp as update_t3_13_0_, user0_.update
2020-08-27 16:27:32.408 [DEBUG] - Exit: eu.ibagroup.easyrpa.cs.service.impl.DataStoreServiceImpl.getByName() with output = DataStoreDto(id=1000, name=product, tableName=ds_3f962e9f1a1b4ed0, descri
2020-08-27 16:27:32.409 [DEBUG] - Enter: eu.ibagroup.easyrpa.cs.service.impl.DataStoreServiceImpl.countRecords() with argument[s] = [1000]
2020-08-27 16:27:32.417 [DEBUG] - Exit: eu.ibagroup.easyrpa.cs.service.impl.DataStoreServiceImpl.countRecords() with output = 108
2020-08-27 16:27:32.800 [DEBUG] - Enter: eu.ibagroup.easyrpa.cs.service.impl.SecretVaultServiceImpl.findByAlias() with argument[s] = [invoiceplane.secrets]
2020-08-27 16:27:32.803 [DEBUG] - Exit: eu.ibagroup.easyrpa.cs.service.impl.SecretVaultServiceImpl.findByAlias() with output = SecretVaultDto(alias=invoiceplane.secrets, value={"user": "admin@workfusion.
2020-08-27 16:27:41.227 [INFO] - Opened connection [connectionId{localValue:2, serverValue:93}] to 10.224.0.35:27017
2020-08-27 16:27:41.247 [INFO] - No details history for ebd8649e-989b-4b28-9d88-42f6db18e961
2020-08-27 16:27:41.247 [INFO] - 2020-08-27 16:27:41,247 INFO  e.i.e.e.s.i.ExecutionHistoryServiceImpl [main] [ebd8649e-989b-4b28-9d88-42f6db18e961] [] - No details history for ebd8649e-989b-4b28-9
2020-08-27 16:27:41.616 [DEBUG] - Checking that DS exist ....
2020-08-27 16:27:41.617 [DEBUG] - https://10.224.0.35/api/v1/ds/byName/product
2020-08-27 16:27:41.617 [INFO] - 2020-08-27 16:27:41,616 DEBUG e.i.e.p.impl.DataStoreUtils [main] [ebd8649e-989b-4b28-9d88-42f6db18e961] [] - Checking that DS exist ....
2020-08-27 16:27:41.618 [INFO] - 2020-08-27 16:27:41,617 DEBUG e.i.e.engine.cs.api.GenericCsClient [main] [ebd8649e-989b-4b28-9d88-42f6db18e961] [] - https://10.224.0.35/api/v1/ds/byName/product
2020-08-27 16:27:42.054 [DEBUG] - DataStore result: DataStoreDto(id=1000, name=product, description=product, schema=eu.ibagroup.easyrpa.engine.cs.api.dto.DataStoreSchema@6edcad64)
2020-08-27 16:27:42.079 [DEBUG] - DS exist ....
2020-08-27 16:27:42.079 [INFO] - 2020-08-27 16:27:42,079 DEBUG e.i.e.p.impl.DataStoreUtils [main] [ebd8649e-989b-4b28-9d88-42f6db18e961] [] - DS exist ....
2020-08-27 16:27:42.079 [INFO] - 2020-08-27 16:27:42,054 DEBUG e.i.e.p.impl.DataStoreUtils [main] [ebd8649e-989b-4b28-9d88-42f6db18e961] [] - DataStore result: DataStoreDto(id=1000, name=product
```

(v. 2.2) Best Practices

The Best Practices section describes techniques that considered a standard way of doing things and allows to maintain quality of the code developed.

Avoid using of Thread.sleep

You should always avoid using of Thread.sleep() method during the RPA implementation because it stops executing the thread during specified time.

Instead of that, you should use driver wait object when you need to wait for some UI element to be appeared on the screen. Driver wait is flexible tool for control of waiting, ignoring exception, control of pollling interval and etc.

E.g.:

Instead of:

Bad practice

```
driver.get("...");
Thread.sleep(2000);
driver.find(By.id("id1")).click();
Thread.sleep(2000);
```

You should do:

Best practice

```
driver.get("...");
WebDriverWait wait = new WebDriverWait(driver, 10);
wait.until(ExpectedConditions.presenceOfElementLocated(By.id("id1")));
driver.find(By.id("id1")).click();
```

Set default driver implicitly wait to 0 seconds

We recommend to always set driver implicitly wait to 0 seconds. As per the documentation of Explicit and Implicit Waits it is clearly mentioned that:

Warning

Do not mix implicit and explicit waits. Doing so can cause unpredictable wait times. For example, setting an implicit wait of 10 seconds and an explicit wait of 15 seconds could cause a timeout to occur after 20 seconds.

Part of the problem is that implicit waits are often (but may not always be!) implemented on the "remote" side of the Driver system. That means they're "baked in" to chromedriver.exe extension that gets installed. Explicit waits are implemented exclusively in the "local" language bindings.

Best practice

```
driver.manage().timeouts().implicitlyWait(0, TimeUnit.SECONDS)
WebDriverWait wait = new WebDriverWait(driver, 10);
wait.until(ExpectedConditions.presenceOfElementLocated(By.id("id1")));
```

Use Page Object design pattern

As common development practice it is recommended to separate business logic from the logic of page of some UI application. In case of RPA, Page Object design pattern is a great fit. Detailed information is available under Page Object Design and Application page.

(v. 2.2) RPA Utils

This is a useful library available in easyRPA nexus by:

```
<dependency>
    <groupId>eu.ibagroup</groupId>
    <artifactId>easy-rpa-utils</artifactId>
    <version>2.0</version>
</dependency>
```

(v. 2.2) Email Library

Email-utils is a set of utilities for working with various e-mail protocols. It allows you to send and receive e-mails as well as manage mailboxes using the following protocols:

- **SMTP**
- **POP3**
- **Exchane**

Below there are three examples of how to use this library.

Important

In this article many parameters and credantials are declared in code. It is highly recommended to use ConfigurationService and VaultService to work with it in projects.

1. **Send a simple email via SMTP.**

SendSimpleEmailTask.java

```java
package eu.ibagroup.easyrpa.email.task;

import eu.ibagroup.easyrpa.email.RobotEmail;
import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApTaskBase;
import eu.ibagroup.easyrpa.engine.model.SecretCredentials;
import lombok.extern.slf4j.Slf4j;

@ApTaskEntry(name = "Send Email using EmailUtils")
@Slf4j
public class SendSimpleEmailTask extends ApTaskBase {

    private static final String RECIPIENTS = "example@gmail.com";
    private static final String SUBJECT = "Test email";
    private static final String EMAIL_SERVICE = "smtp.gmail.com:587"; //gmail service for example
    private static final String EMAIL_SERVICE_PROTOCOL = "smtp_over_tsl";
    private static final String SENDER_NAME = "EasyRPA Bot";
    private static final String BODY = "This message was sent by EasyRPA Bot.";

    @Override
    public void execute() {

        SecretCredentials secret = new SecretCredentials("botmailbox@gmail.com", "password");

        RobotEmail emailSender = new RobotEmail();
        emailSender.setCredentials(secret);
        emailSender.setEmailService(EMAIL_SERVICE);
        emailSender.setEmailServiceProtocol(EMAIL_SERVICE_PROTOCOL);
        emailSender.setSender(secret.getUser());

        emailSender
                .subject(SUBJECT)
                .recipients(RECIPIENTS)
                .body(BODY)
                .setSenderName(SENDER_NAME);
        emailSender.send();
    }
}
```

2. **Generate a table in the body of the message using ftl-template, css and send via SMTP.**

In this approach, you have to create a robot class that extends RobotEmail, override the beforeSend() method, configure the robot in this method and then call the send() method from its object.

RobotEmail uses FreeMarker to generate the message body code. Read more about FreeMarker and ftl templates **here**.

SendFtlEmailTask.java

```
package eu.ibagroup.easyrpa.email.task;

import eu.ibagroup.easyrpa.email.entities.Book;
import eu.ibagroup.easyrpa.email.robot.SendFtlEmailRobot;
import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApTaskBase;
import eu.ibagroup.easyrpa.engine.model.SecretCredentials;
import lombok.extern.slf4j.Slf4j;

import java.util.Arrays;
import java.util.List;
import java.util.UUID;

@ApTaskEntry(name = "Generate Email Report")
@Slf4j
public class SendFtlEmailTask extends ApTaskBase {

    @Override
    public void execute() {
        try {
            List<Book> books = getBooks();

            SendFtlEmailRobot robot = new SendFtlEmailRobot(new SecretCredentials("botmailbox@gmail.
com", "password"));
            robot.setDebtorsList(books).send();
        } catch (Exception e) {
            handleStepError(e);
        }
    }

    //prepare test data
    private List<Book> getBooks() {
        Book thinkingInJava = new Book(UUID.randomUUID().toString(), "Thinking in Java", "Bruce Eckel");
        Book cipollino = new Book(UUID.randomUUID().toString(), "Le avventure di Cipollino", "Giovanni
Francesco Rodari");
        Book wadAndPeace = new Book(UUID.randomUUID().toString(), "War and Peace", "Lev Tolstoy");

        List<Book> books = Arrays.asList(thinkingInJava, cipollino, wadAndPeace);
        return books;
    }
}
```

SendFtlEmailRobot.java

```
package eu.ibagroup.easyrpa.email.robot;

import com.google.common.io.Resources;
import eu.ibagroup.easyrpa.email.RobotEmail;
import eu.ibagroup.easyrpa.email.entities.Book;
import eu.ibagroup.easyrpa.engine.model.SecretCredentials;

import java.io.IOException;
import java.net.URL;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.List;

public class SendFtlEmailRobot extends RobotEmail {

    private static final String TEMPLATE_FILE_PATH = "books.email.tpl/books.ftl";
    private static final String RECIPIENTS = "example@gmail.com";
    private static final String SUBJECT = "Book list";
    private static final String EMAIL_SERVICE = "smtp.gmail.com:587"; //gmail service for example
    private static final String EMAIL_SERVICE_PROTOCOL = "smtp_over_tsl";
    private static final String SENDER_NAME = "EasyRPA Bot";

    //typeName is a prefix for the parameters of a specific robot in the configuration file
    private static final String TYPE_NAME = "books_email";

    private List<Book> books;
    private SecretCredentials secret;

    public SendFtlEmailRobot(SecretCredentials secret) {
        super();
        this.secret = secret;
        setTypeName(TYPE_NAME);
    }

    public SendFtlEmailRobot setDebtorsList(List<Book> books) {
        this.books = books;
        return this;
    }

    @Override
    protected void beforeSend() {
        setRecipients(Arrays.asList(RECIPIENTS));
        setCredentials(secret);
        subject(SUBJECT);
        setEmailService(EMAIL_SERVICE);
        setEmailServiceProtocol(EMAIL_SERVICE_PROTOCOL);
        sender(secret.getUser());
        senderName(SENDER_NAME);

        URL url = Resources.getResource(TEMPLATE_FILE_PATH);
        String body = null;
        try {
            body = Resources.toString(url, StandardCharsets.UTF_8);
        } catch (IOException e) {
            //do something
        }
        body(body);
        property("books", books);
    }
}
```

The source code of the template and css:

books.ftl

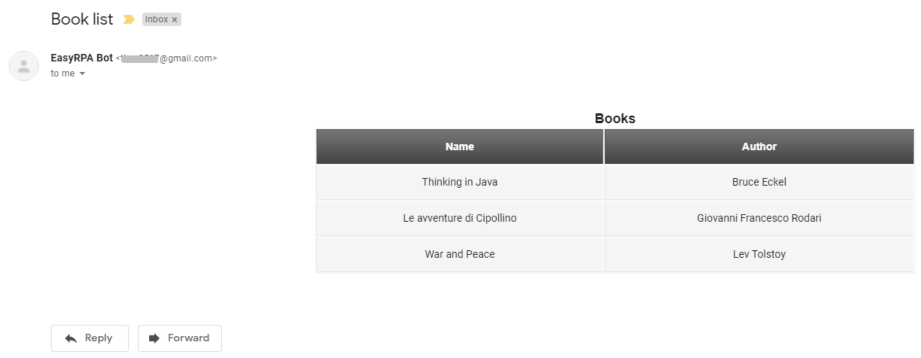
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <#include "books.email.tpl/books.css">
</head>
<body>

<div class="container">
  <table class="responsive-table">
    <caption>Books</caption>
    <thead>
      <tr>
        <th scope="col">Name</th>
        <th scope="col">Author</th>
      </tr>
    </thead>
    <tbody>
```

```
      <#list books as book>
      <tr>
        <td data-title="Name">${book.name}</td>
        <td data-title="Author">${book.author}</td>
      </tr>
      </#list>
    </tbody>
  </table>
</div>

</body>
</html>
```

books.css

```
<style>
body {
        background: #fafafa url(https://jackrugile.com/images/misc/noise-diagonal.png);
        color: #444;
        font: 100%/30px 'Helvetica Neue', helvetica, arial, sans-serif;
        text-shadow: 0 1px 0 #fff;
}

strong {
        font-weight: bold;
}

em {
        font-style: italic;
}

table {
        background: #f5f5f5;
        border-collapse: separate;
        box-shadow: inset 0 1px 0 #fff;
        font-size: 14px;
        line-height: 24px;
        margin: 30px auto;
        text-align: center;
        width: 800px;
}

caption {
     font-size: 18px;
     font-weight: bold;
     }

th {
        background: url(https://jackrugile.com/images/misc/noise-diagonal.png), linear-gradient(#777,
#444);
        border-left: 1px solid #555;
        border-right: 1px solid #777;
        border-top: 1px solid #555;
        border-bottom: 1px solid #333;
        box-shadow: inset 0 1px 0 #999;
        color: #fff;
  font-weight: bold;
        padding: 10px 15px;
        position: relative;
        text-shadow: 0 1px 0 #000;
}

th:after {
        background: linear-gradient(rgba(255,255,255,0), rgba(255,255,255,.08));
        content: '';
        display: block;
        height: 25%;
        left: 0;
        margin: 1px 0 0 0;
        position: absolute;
        top: 25%;
        width: 100%;
}

th:first-child {
        border-left: 1px solid #777;
        box-shadow: inset 1px 1px 0 #999;
}

th:last-child {
        box-shadow: inset -1px 1px 0 #999;
}

td {
        border-right: 1px solid #fff;
        border-left: 1px solid #e8e8e8;
        border-top: 1px solid #fff;
        border-bottom: 1px solid #e8e8e8;
        padding: 10px 15px;
        position: relative;
        transition: all 300ms;
}

td:first-child {
        box-shadow: inset 1px 0 0 #fff;
}

td:last-child {
        border-right: 1px solid #e8e8e8;
        box-shadow: inset -1px 0 0 #fff;
}

tr {
        background: url(https://jackrugile.com/images/misc/noise-diagonal.png);
}

tr:nth-child(odd) td {
        background: #f1f1f1 url(https://jackrugile.com/images/misc/noise-diagonal.png);
}

tr:last-of-type td {
        box-shadow: inset 0 -1px 0 #fff;
}

tr:last-of-type td:first-child {
        box-shadow: inset 1px -1px 0 #fff;
}

tr:last-of-type td:last-child {
        box-shadow: inset -1px -1px 0 #fff;
}

tbody:hover td {
        color: transparent;
        text-shadow: 0 0 3px #aaa;
}

tbody:hover tr:hover td {
        color: #444;
        text-shadow: 0 1px 0 #fff;
}

</style>
```

Result Email:



Book list

EasyRPA Bot <[redacted]@gmail.com>

Books

Name	Author
Thinking in Java	Bruce Eckel
Le avventure di Cipollino	Giovanni Francesco Rodari
War and Peace	Lev Tolstoy

3. **Receive mails and mailbox folder list via POP3 protocol.**

ReadEmailTask.java

```
package eu.ibagroup.easyrpa.email.task;

import eu.ibagroup.easyrpa.email.EmailClientProvider;
import eu.ibagroup.easyrpa.email.message.EmailMessage;
import eu.ibagroup.easyrpa.email.service.javax.ImapPop3EmailClient;
import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.engine.model.SecretCredentials;
import eu.ibagroup.easyrpa.engine.service.ConfigurationService;
import lombok.extern.slf4j.Slf4j;

import javax.mail.search.AndTerm;
import javax.mail.search.ComparisonTerm;
import javax.mail.search.ReceivedDateTerm;
import javax.mail.search.SearchTerm;

import java.util.Calendar;
import java.util.Date;
import java.util.List;

@ApTaskEntry(name = "Read Email using EmailUtils")
@Slf4j
public class ReadEmailTask extends ApTask {

        private static final String EMAIL_SERVICE = "imap.gmail.com:993";

        private static final String EMAIL_SERVICE_PROTOCOL = "imap";

        @Override
        public void execute() {
                ConfigurationService cfg = getConfigurationService();
                SecretCredentials credentials = new SecretCredentials("example@gmail.com", "password");

                EmailClientProvider emailScanner = new EmailClientProvider(cfg, credentials);
                emailScanner.setEmailService(EMAIL_SERVICE);
                emailScanner.setEmailServiceProtocol(EMAIL_SERVICE_PROTOCOL);

                ImapPop3EmailClient client = (ImapPop3EmailClient) emailScanner.getClient();

                // Now you can implement any search logic and call ImapPop3EmailClient method
                // e.g.
                // Get list of folders
                log.info("List of folders: " + client.fetchFolderList().toString());
                // Fetch all messages from the 'INBOX' folder during the last 3 days
                Date currentDate = new Date();
                Calendar cal = Calendar.getInstance();
                cal.setTime(currentDate);
                cal.add(Calendar.DATE, -3);
                Date fromDate = cal.getTime();

                cal.setTime(currentDate);
                cal.add(Calendar.DATE, 1);
                Date toDate = cal.getTime();
                SearchTerm olderThanTerm = new ReceivedDateTerm(ComparisonTerm.LE, toDate);
                SearchTerm newerThanTerm = new ReceivedDateTerm(ComparisonTerm.GE, fromDate);

                SearchTerm dateRangeTerm = new AndTerm(olderThanTerm, newerThanTerm);

                List<EmailMessage> messages = client.searchMessages("INBOX", dateRangeTerm);
                for (EmailMessage msg : messages) {
                        log.info(msg.getSubject());
                }

        }
}
```

Other project files required to launch the example:

SendEmailsAp

```
package eu.ibagroup.easyrpa.email;

import eu.ibagroup.easyrpa.email.task.ReadEmailTask;
import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModule;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;

@ApModuleEntry(name = "Test EmailUtils")
public class SendEmailsAp extends ApModule {

        // change ReadEmail.class to any of the two remaining task-classes
        public TaskOutput run() throws Exception {
                return execute(getInput(), ReadEmailTask.class).get();
        }
}
```

LocalRunner.java

```
package eu.ibagroup.easyrpa.email;

import eu.ibagroup.easyrpa.engine.boot.ApModuleRunner;

public class LocalRunner {

        public static void main(String[] args) {
                ApModuleRunner.localLaunch(SendEmailsAp.class);
        }
}
```

(v. 2.2) Excel Library

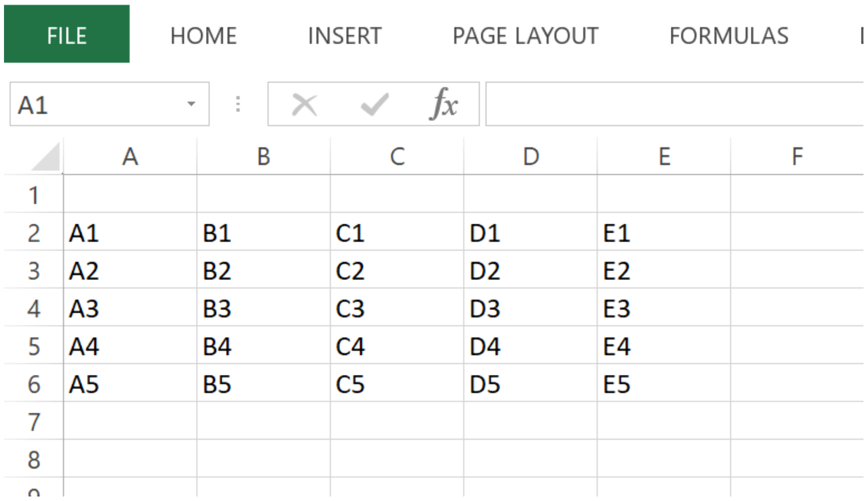
Excel-utils is a set of utilities for working with excel spreadsheets. It is based on the Apache POI library and the cscript utility for running vba scripts.

Below is a small example of working with an existing document. Here we create a sheet and fill it with data.

ExcelTask.java

```
package eu.ibagroup.easyrpa.excel.task;

import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.excel.SpreadsheetDocument;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.util.CellReference;

import java.io.FileInputStream;

@ApTaskEntry(name = "Create and read excel file")
public class ExcelTask extends ApTask {

        private static final String FILE_PATH = "app/test.xlsx";

        @Override
        public void execute() {
                try {
                        SpreadsheetDocument doc = new SpreadsheetDocument(new FileInputStream(FILE_PATH));
                        Sheet numbersSheet = doc.createSheet("Numbers");

                        for (int i = 0; i < 5; i++) {
                                Row row = numbersSheet.createRow(i);
                                for (int j = 0; j < 5; j++) {
                                        Cell cell = row.createCell(j);
                                        cell.setCellValue(String.format("%s%d", CellReference.
convertNumToColString(j), i + 1));
                                }
                        }

                        doc.writeToFile(FILE_PATH);
                } catch (Exception e) {
                        e.printStackTrace();
                        // do something
                }
        }
}
```

Result sheet:



	A	B	C	D	E	F
1						
2	A1	B1	C1	D1	E1	
3	A2	B2	C2	D2	E2	
4	A3	B3	C3	D3	E3	
5	A4	B4	C4	D4	E4	
6	A5	B5	C5	D5	E5	
7						
8						

ExcelSampleAp.java

```
package eu.ibagroup.easyrpa.excel;

import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModule;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import eu.ibagroup.easyrpa.excel.task.ExcelTask;

@ApModuleEntry(name = "Test ExcelUtils")
public class ExcelSampleAp extends ApModule {
        public TaskOutput run() throws Exception {
                return execute(getInput(), ExcelTask.class).get();
        }
}
```

LocalRunner.java

```
package eu.ibagroup.easyrpa.excel;

import eu.ibagroup.easyrpa.engine.boot.ApModuleRunner;

public class LocalRunner {
        public static void main(String[] args) {
                ApModuleRunner.localLaunch(ExcelSampleAp.class);
        }
}
```

(v. 2.2) Debug AP on remote node with Control Server

Overview

In many cases when you can not setup client applications on developer machine, it could be possible to check automation run on remote node. Node runs Automation Process as separate java process, so we can use standard Java debugging approach for this.

Specify debug flags on node

Go to the node configuration and setup required parameters:

Name	Parameter	Default value
JAVA_DEBUG	Enable or disable AP debug	n
JAVA_DEBUG_PORT_BASE	Base debug port. Since several JVM can be run on node this defines base port for debugging. If port is busy Node will find first available port.	1044
JAVA_DEBUG_SUSPEND	Suspend target JVM	n
JAVA_DEBUG_HOST	Host to listen on. localhost by default	

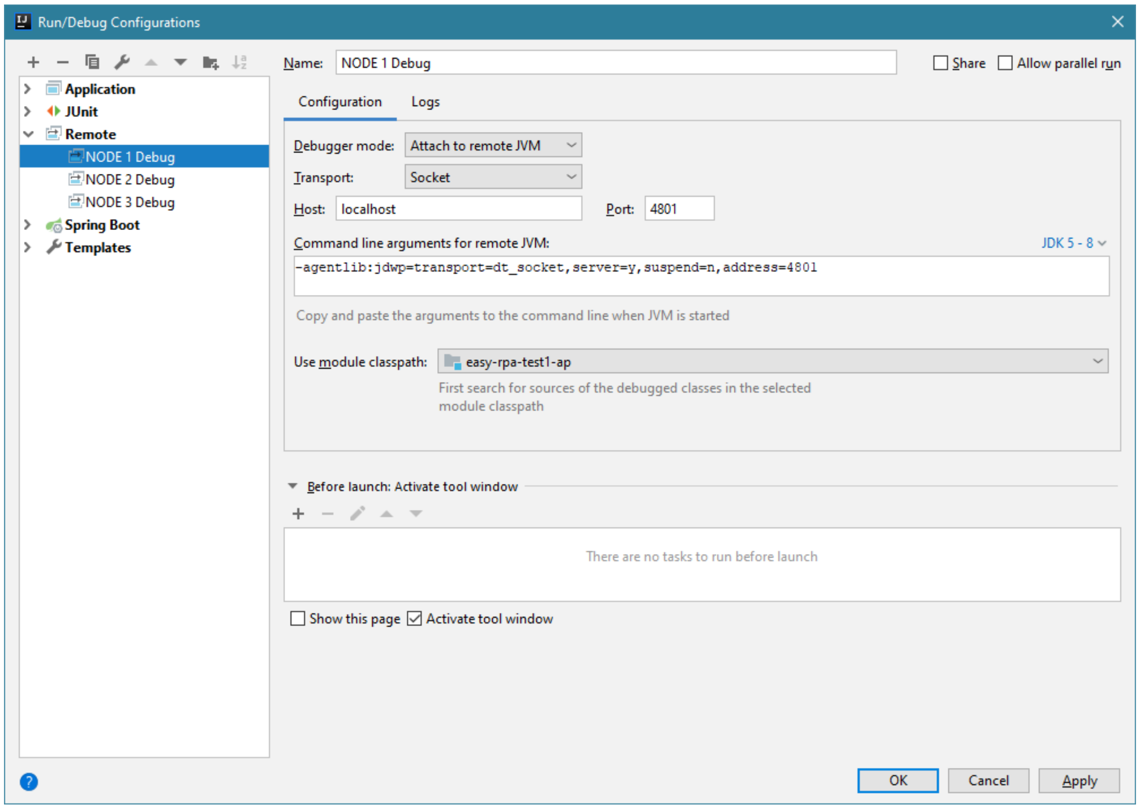
In AP log following line will appear

-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=**1050**

Address parameter is calculated port number debugger listening on.

Connect remote debugger

In your IDE setup remote java debug launch configuration. For IDEA it looks like:



Name: NODE 1 Debug

Debugger mode: Attach to remote JVM

Transport: Socket

Host: localhost Port: 4801

Command line arguments for remote JVM: JDK 5 - 8

```
-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=4801
```

Copy and paste the arguments to the command line when JVM is started

Use module classpath: easy-rpa-test1-ap

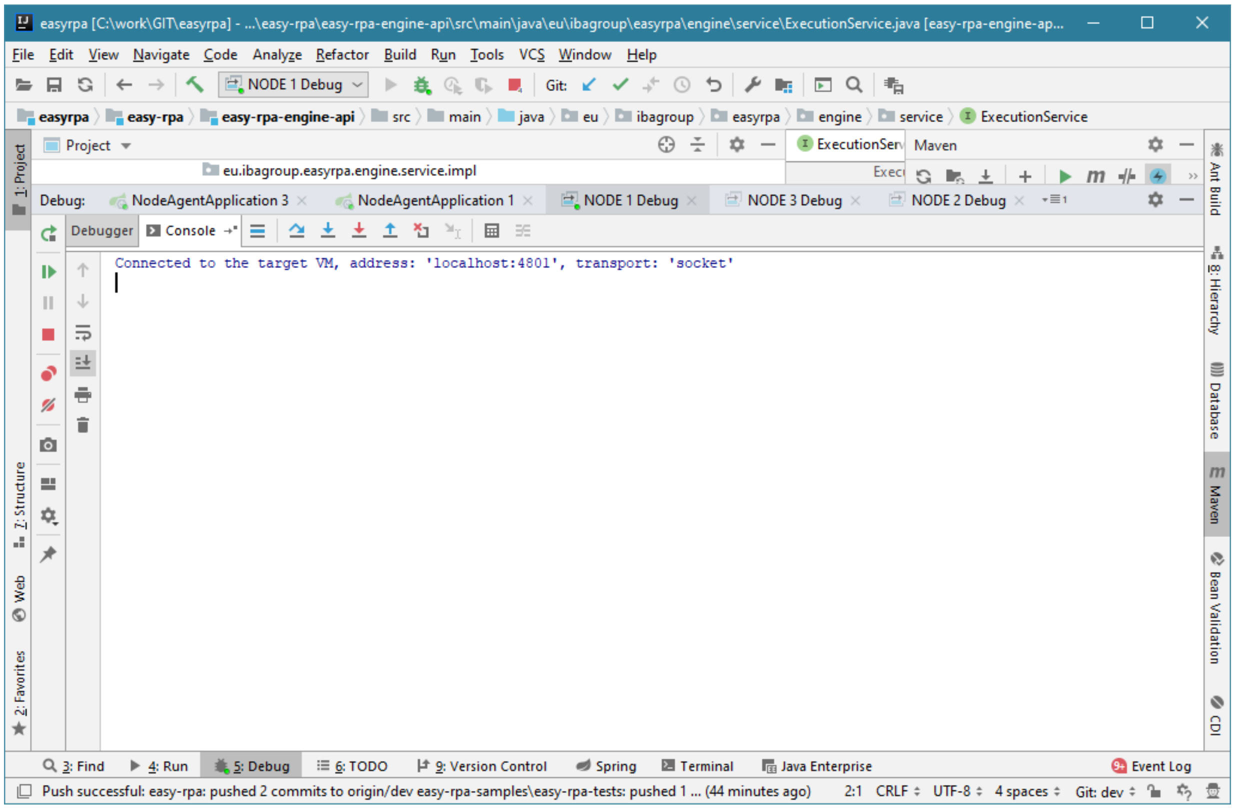
First search for sources of the debugged classes in the selected module classpath

Before launch: Activate tool window

There are no tasks to run before launch

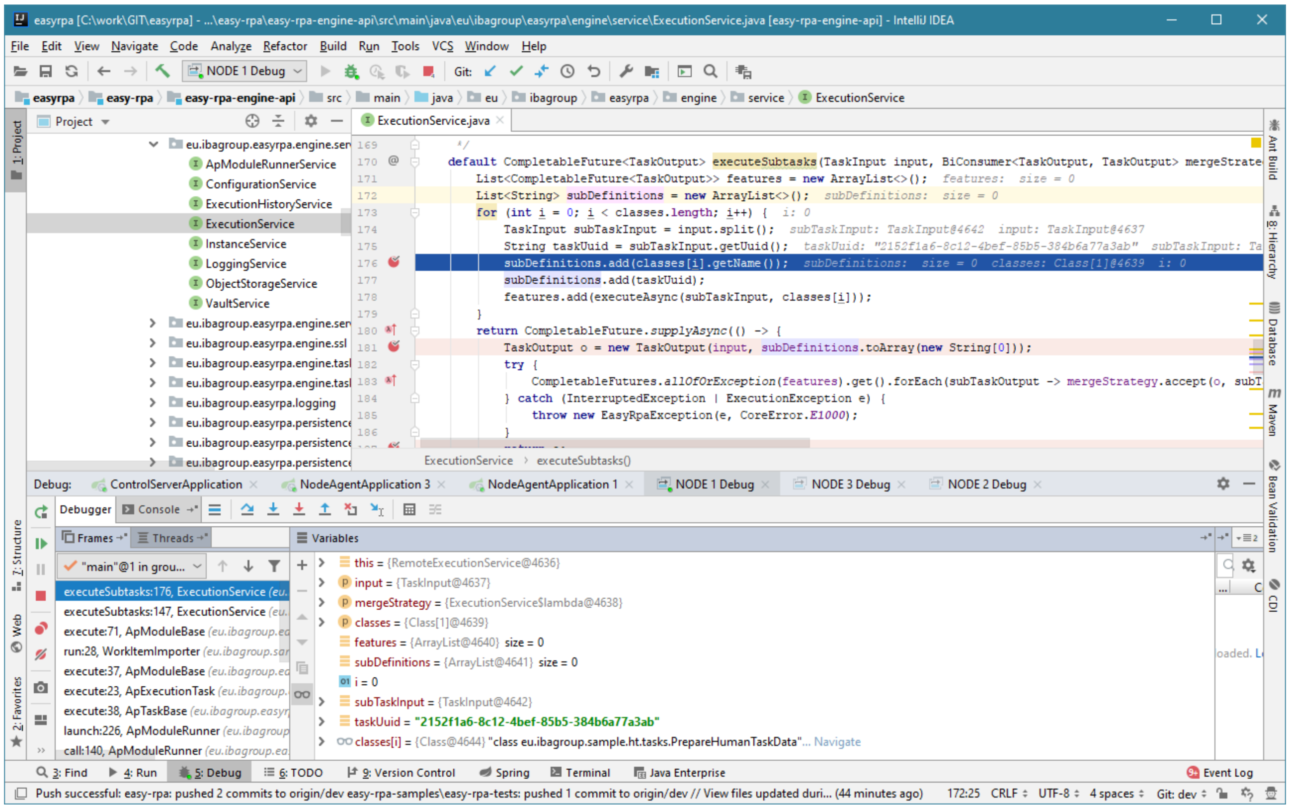
Show this page ☑ Activate tool window

Observe node logs and as soon as the message of connection waiting appears, launch the debugger configuration:


easyrpa [C:\work\GIT\easyrpa] - ...\easy-rpa\easy-rpa-engine-api\src\main\java\eu\ibagroup\easyrpa\engine\service\ExecutionService.java [easy-rpa-engine-ap...
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
NODE 1 Debug
Git:
easyrpa easy-rpa easy-rpa-engine-api src main java eu ibagroup easyrpa engine service ExecutionService
Project
ExecutionServ
Maven
eu.ibagroup.easyrpa.engine.service.impl
Debug: NodeAgentApplication 3 NodeAgentApplication 1 NODE 1 Debug NODE 3 Debug NODE 2 Debug
Debugger
Console
1: Project
7: Structure
Web
2: Favorites
Ant Build
8: Hierarchy
Database
Maven
Bean Validation
CDI
3: Find 4: Run 5: Debug 6: TODO 9: Version Control Spring Terminal Java Enterprise Event Log

```
Connected to the target VM, address: 'localhost:4801', transport: 'socket'
```

Push successful: easy-rpa: pushed 2 commits to origin/dev easy-rpa-samples\easy-rpa-tests: pushed 1 ... (44 minutes ago) 2:1 CRLF UTF-8 4 spaces Git: dev



easyrpa [C:\work\GIT\easyrpa] - ...\easy-rpa\easy-rpa-engine-api\src\main\java\eu\ibagroup\easyrpa\engine\service\ExecutionService.java [easy-rpa-engine-api] - IntelliJ IDEA

```
*/
default CompletableFuture<TaskOutput> executeSubtasks(TaskInput input, BiConsumer<TaskOutput, TaskOutput> mergeStrate
    List<CompletableFuture<TaskOutput>> features = new ArrayList<>();  features:  size = 0
    List<String> subDefinitions = new ArrayList<>();  subDefinitions:  size = 0
    for (int i = 0; i < classes.length; i++) {  i: 0
        TaskInput subTaskInput = input.split();  subTaskInput: TaskInput@4642  input: TaskInput@4637
        String taskUuid = subTaskInput.getUuid();  taskUuid: "2152f1a6-8c12-4bef-85b5-384b6a77a3ab"  subTaskInput: Ta
        subDefinitions.add(classes[i].getName());  subDefinitions:  size = 0  classes: Class[1]@4639  i: 0
        subDefinitions.add(taskUuid);
        features.add(executeAsync(subTaskInput, classes[i]));
    }
    return CompletableFuture.supplyAsync(() -> {
        TaskOutput o = new TaskOutput(input, subDefinitions.toArray(new String[0]));
        try {
            CompletableFutures.allOfOrException(features).get().forEach(subTaskOutput -> mergeStrategy.accept(o, subT
        } catch (InterruptedException | ExecutionException e) {
            throw new EasyRpaException(e, CoreError.E1000);
        }
```

```
executeSubtasks:176, ExecutionService (eu.
executeSubtasks:147, ExecutionService (eu.
execute:71, ApModuleBase (eu.ibagroup.ea
run:28, WorkItemImporter (eu.ibagroup.sar
execute:37, ApModuleBase (eu.ibagroup.ea
execute:23, ApExecutionTask (eu.ibagroup.
execute:38, ApTaskBase (eu.ibagroup.easyr
launch:226, ApModuleRunner (eu.ibagroup
call:140, ApModuleRunner (eu.ibagroup.ea
```

```
this = {RemoteExecutionService@4636}
input = {TaskInput@4637}
mergeStrategy = {ExecutionService$lambda@4638}
classes = {Class[1]@4639}
features = {ArrayList@4640}  size = 0
subDefinitions = {ArrayList@4641}  size = 0
i = 0
subTaskInput = {TaskInput@4642}
taskUuid = "2152f1a6-8c12-4bef-85b5-384b6a77a3ab"
classes[i] = {Class@4644} "class eu.ibagroup.sample.ht.tasks.PrepareHumanTaskData"... Navigate
```

Push successful: easy-rpa: pushed 2 commits to origin/dev easy-rpa-samples\easy-rpa-tests: pushed 1 commit to origin/dev // View files updated duri... (44 minutes ago)

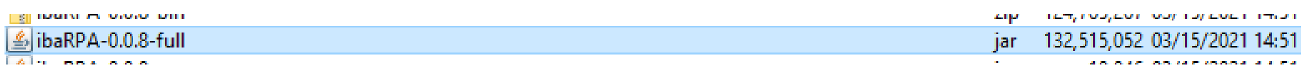
(v. 2.2) Debug AP on remote node without Control Server

Sometimes RPA developer can start implementation earlier than EasyRPA Control Server and infrastructure are prepared. It's possible to run Automation Process on remote node. Node runs Automation Process as separate java process, so we can use standard Java debugging approach for this.

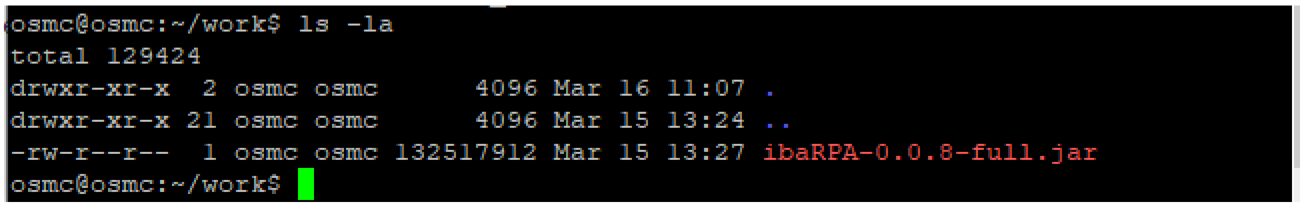
1. Build "uber" jar of the automation process project, for example using "shade" plugin:

pom.xml

```
<plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-shade-plugin</artifactId>
        <configuration>
                <filters>
                        <filter>
                                <artifact>*:*</artifact>
                                <excludes>
                                        <exclude>module-info.class</exclude>
                                        <exclude>META-INF/*.SF</exclude>
                                        <exclude>META-INF/*.DSA</exclude>
                                        <exclude>META-INF/*.RSA</exclude>
                                </excludes>
                        </filter>
                </filters>
                <transformers>
                        <transformer
                                implementation="org.apache.maven.plugins.shade.resource.
ManifestResourceTransformer">
                                <mainClass>eu.ibagroup.easyrpa.engine.boot.ApModuleRunner</mainClass>
                        </transformer>
                </transformers>
        </configuration>
        <executions>
                <execution>
                <goals>
                        <goal>shade</goal>
                </goals>
                <configuration>
                        <shadedArtifactAttached>true</shadedArtifactAttached>
                        <shadedClassifierName>full</shadedClassifierName>
                        <transformers>
                                <transformer
                                        implementation="org.apache.maven.plugins.shade.resource.
ManifestResourceTransformer">
                                </transformer>
                        </transformers>
                </configuration>
                </execution>
        </executions>
</plugin>
```


ibaRPA-0.0.8-full jar 132,515,052 03/15/2021 14:51

2. Upload the jar to remote machine. In the example below it was uploaded to Linux Node Agent using ssh:



```
osmc@osmc:~/work$ ls -la
total 129424
drwxr-xr-x  2 osmc osmc      4096 Mar 16 11:07 .
drwxr-xr-x 21 osmc osmc      4096 Mar 15 13:24 ..
-rw-r--r--  1 osmc osmc 132517912 Mar 15 13:27 ibaRPA-0.0.8-full.jar
osmc@osmc:~/work$
```

3. Install JRE on the remote machine.

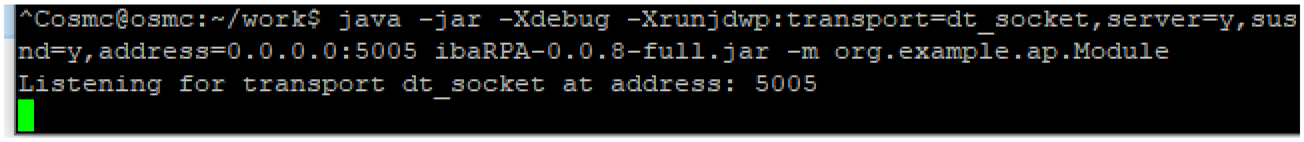
4. Launch the Automation Process jar as remote server specifying debug flag:

Command

```
java -jar -Xdebug -Xrunjdwp:transport=dt_socket,server=y,suspend=y,address=0.0.0.0:5005 ibaRPA-0.0.8-full.jar -
m org.example.ap.Module
```

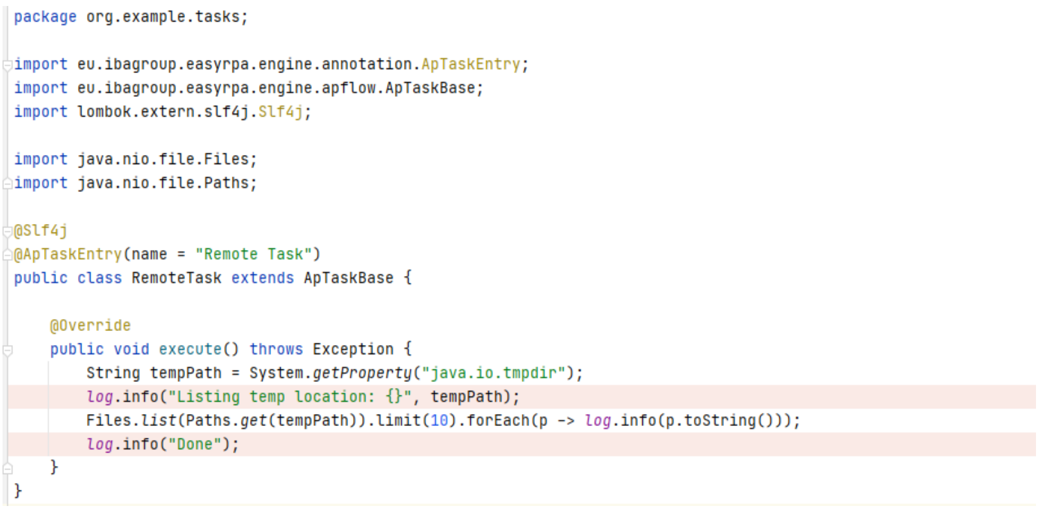
where "-m" - module to launch

As result, you should see message "Listening for transport dt_socket at address : 5005":



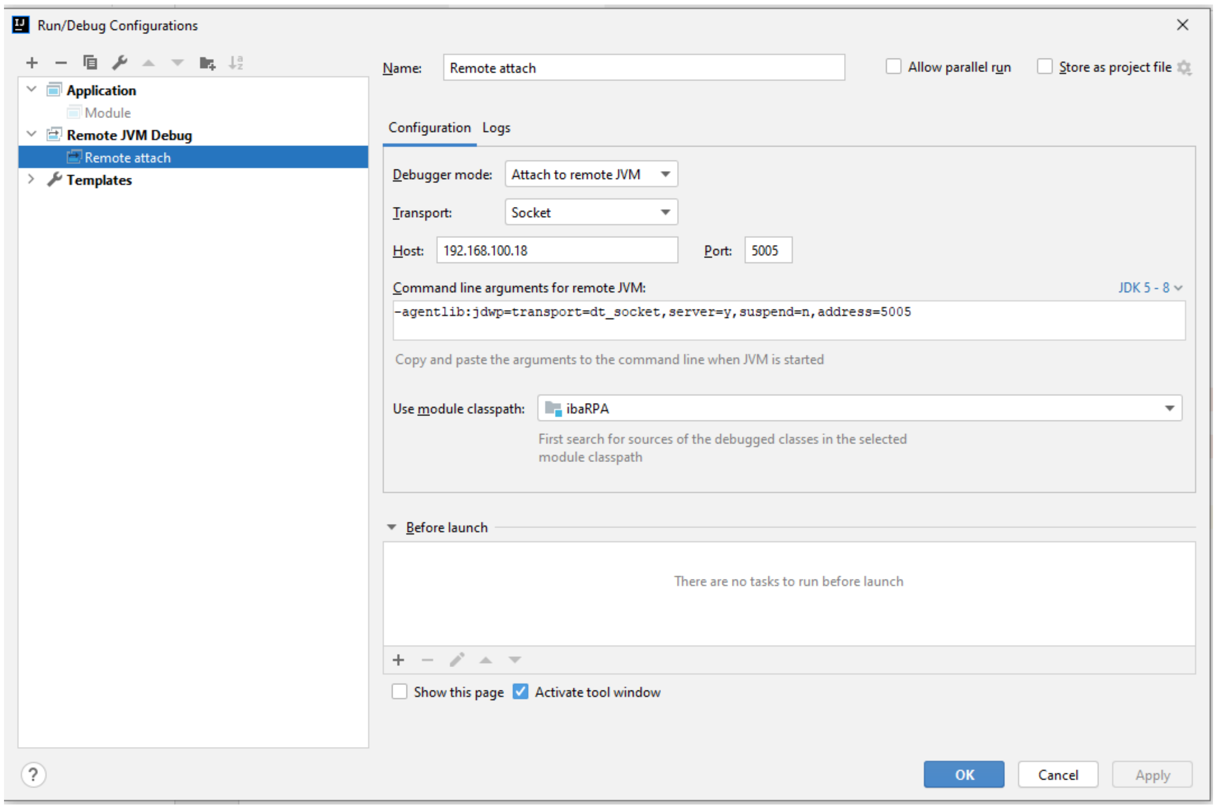
```
^Cosmc@osmc:~/work$ java -jar -Xdebug -Xrunjdwp:transport=dt_socket,server=y,sus
nd=y,address=0.0.0.0:5005 ibaRPA-0.0.8-full.jar -m org.example.ap.Module
Listening for transport dt_socket at address: 5005
```

5. On local machine in Intellij IDEA add breakpoint in a Task:



```java
package org.example.tasks;

import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApTaskBase;
import lombok.extern.slf4j.Slf4j;

import java.nio.file.Files;
import java.nio.file.Paths;

@Slf4j
@ApTaskEntry(name = "Remote Task")
public class RemoteTask extends ApTaskBase {

    @Override
    public void execute() throws Exception {
        String tempPath = System.getProperty("java.io.tmpdir");
        log.info("Listing temp location: {}", tempPath);
        Files.list(Paths.get(tempPath)).limit(10).forEach(p -> log.info(p.toString()));
        log.info("Done");
    }
}
```

6. Create new Run Configuration Remote JVM Debug:



Run/Debug Configurations

- Application
  - Module
- Remote JVM Debug
  - Remote attach
- Templates

Name: Remote attach

☐ Allow parallel run ☐ Store as project file

Configuration Logs

Debugger mode: Attach to remote JVM

Transport: Socket

Host: 192.168.100.18 Port: 5005

Command line arguments for remote JVM: JDK 5 - 8

```
-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=5005
```

Copy and paste the arguments to the command line when JVM is started

Use module classpath: ibaRPA

First search for sources of the debugged classes in the selected module classpath

Before launch

There are no tasks to run before launch

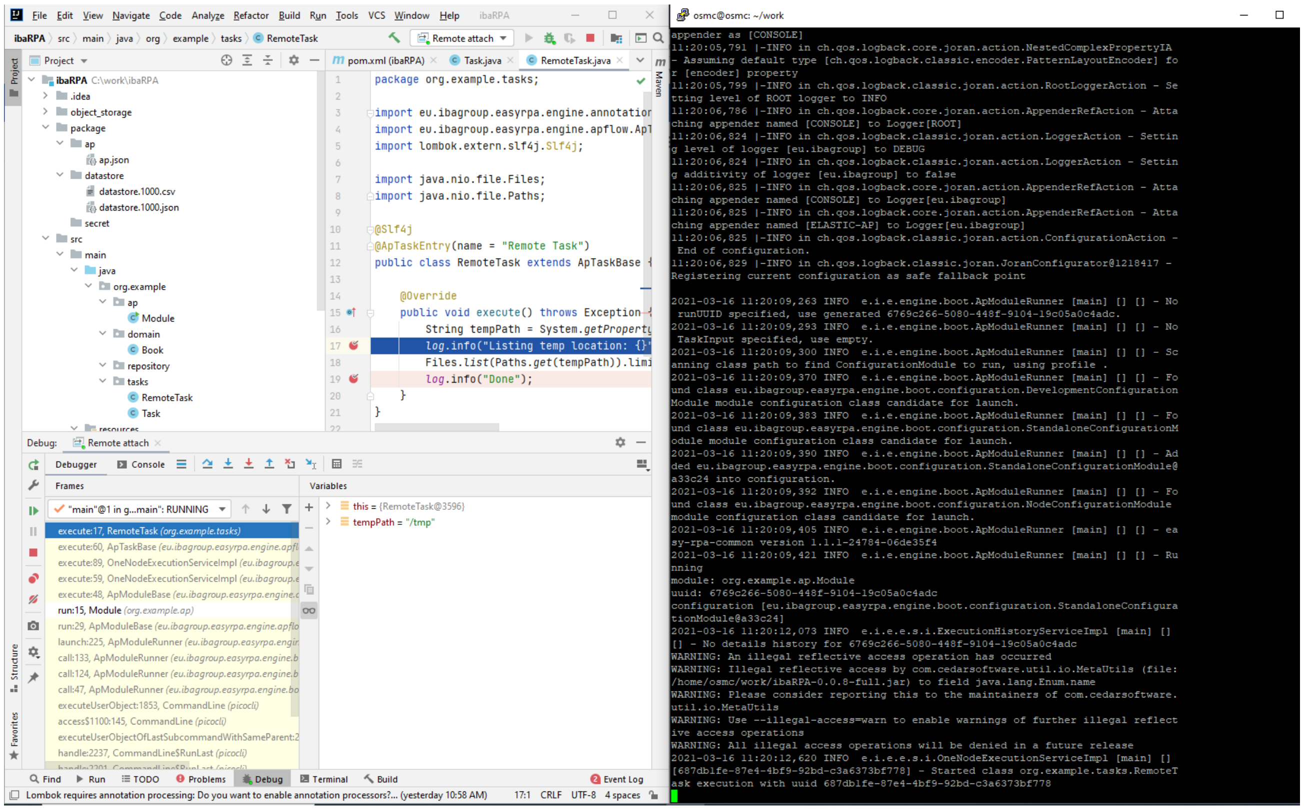
☐ Show this page ☑ Activate tool window

OK Cancel Apply

Select "Attach to remote JVM " option under "Debugger mode";

Specify Host address and Port.

7.Launch the Run Configuration and wait until debugger is connected / suspend on the first break point:



```
package org.example.tasks;

import eu.ibagroup.easyrpa.engine.annotatior
import eu.ibagroup.easyrpa.engine.apflow.ApT
import lombok.extern.slf4j.Slf4j;

import java.nio.file.Files;
import java.nio.file.Paths;

@Slf4j
@ApTaskEntry(name = "Remote Task")
public class RemoteTask extends ApTaskBase {

    @Override
    public void execute() throws Exception {
        String tempPath = System.getProperty
        log.info("Listing temp location: {}"
        Files.list(Paths.get(tempPath)).limi
        log.info("Done");
    }
}
```

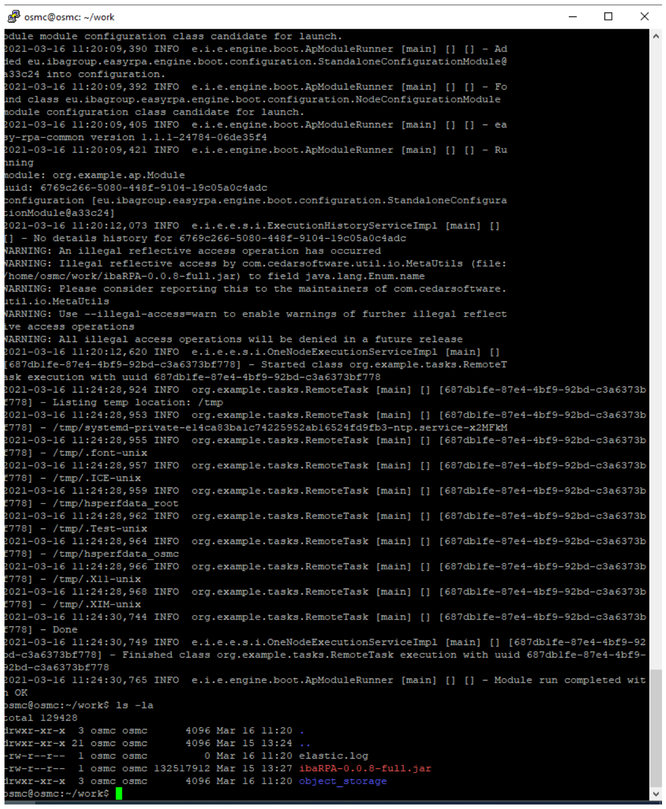
"main"@1 in g...main": RUNNING

```
execute:17, RemoteTask (org.example.tasks)
execute:60, ApTaskBase (eu.ibagroup.easyrpa.engine.apfl
execute:89, OneNodeExecutionServiceImpl (eu.ibagroup.e
execute:59, OneNodeExecutionServiceImpl (eu.ibagroup.e
execute:48, ApModuleBase (eu.ibagroup.easyrpa.engine.a
run:15, Module (org.example.ap)
run:29, ApModuleBase (eu.ibagroup.easyrpa.engine.apflo
launch:225, ApModuleRunner (eu.ibagroup.easyrpa.engin
call:133, ApModuleRunner (eu.ibagroup.easyrpa.engine.b
call:124, ApModuleRunner (eu.ibagroup.easyrpa.engine.b
call:47, ApModuleRunner (eu.ibagroup.easyrpa.engine.bo
executeUserObject:1853, CommandLine (picocli)
access$1100:145, CommandLine (picocli)
executeUserObjectOfLastSubcommandWithSameParent:2
handle:2237, CommandLine$RunLast (picocli)
```

```
this = {RemoteTask@3596}
tempPath = "/tmp"
```

Lombok requires annotation processing: Do you want to enable annotation processors?... (yesterday 10:58 AM)

```
appender as [CONSOLE]
11:20:05,791 |-INFO in ch.qos.logback.core.joran.action.NestedComplexPropertyIA
- Assuming default type [ch.qos.logback.classic.encoder.PatternLayoutEncoder] fo
r [encoder] property
11:20:05,799 |-INFO in ch.qos.logback.classic.joran.action.RootLoggerAction - Se
tting level of ROOT logger to INFO
11:20:06,786 |-INFO in ch.qos.logback.core.joran.action.AppenderRefAction - Atta
ching appender named [CONSOLE] to Logger[ROOT]
11:20:06,824 |-INFO in ch.qos.logback.classic.joran.action.LoggerAction - Settin
g level of logger [eu.ibagroup] to DEBUG
11:20:06,824 |-INFO in ch.qos.logback.classic.joran.action.LoggerAction - Settin
g additivity of logger [eu.ibagroup] to false
11:20:06,825 |-INFO in ch.qos.logback.core.joran.action.AppenderRefAction - Atta
ching appender named [CONSOLE] to Logger[eu.ibagroup]
11:20:06,825 |-INFO in ch.qos.logback.core.joran.action.AppenderRefAction - Atta
ching appender named [ELASTIC-AP] to Logger[eu.ibagroup]
11:20:06,825 |-INFO in ch.qos.logback.classic.joran.action.ConfigurationAction -
 End of configuration.
11:20:06,829 |-INFO in ch.qos.logback.classic.joran.JoranConfigurator@1218417 -
Registering current configuration as safe fallback point

2021-03-16 11:20:09,263 INFO  e.i.e.engine.boot.ApModuleRunner [main] [] [] - No
 runUUID specified, use generated 6769c266-5080-448f-9104-19c05a0c4adc.
2021-03-16 11:20:09,293 INFO  e.i.e.engine.boot.ApModuleRunner [main] [] [] - No
 TaskInput specified, use empty.
2021-03-16 11:20:09,300 INFO  e.i.e.engine.boot.ApModuleRunner [main] [] [] - Sc
anning class path to find ConfigurationModule to run, using profile .
2021-03-16 11:20:09,370 INFO  e.i.e.engine.boot.ApModuleRunner [main] [] [] - Fo
und class eu.ibagroup.easyrpa.engine.boot.configuration.DevelopmentConfiguration
Module module configuration class candidate for launch.
2021-03-16 11:20:09,383 INFO  e.i.e.engine.boot.ApModuleRunner [main] [] [] - Fo
und class eu.ibagroup.easyrpa.engine.boot.configuration.StandaloneConfigurationM
odule module configuration class candidate for launch.
2021-03-16 11:20:09,390 INFO  e.i.e.engine.boot.ApModuleRunner [main] [] [] - Ad
ded eu.ibagroup.easyrpa.engine.boot.configuration.StandaloneConfigurationModule@
a33c24 into configuration.
2021-03-16 11:20:09,392 INFO  e.i.e.engine.boot.ApModuleRunner [main] [] [] - Fo
und class eu.ibagroup.easyrpa.engine.boot.configuration.NodeConfigurationModule
module configuration class candidate for launch.
2021-03-16 11:20:09,405 INFO  e.i.e.engine.boot.ApModuleRunner [main] [] [] - ea
sy-rpa-common version 1.1.1-24784-06de35f4
2021-03-16 11:20:09,421 INFO  e.i.e.engine.boot.ApModuleRunner [main] [] [] - Ru
nning
module: org.example.ap.Module
uuid: 6769c266-5080-448f-9104-19c05a0c4adc
configuration [eu.ibagroup.easyrpa.engine.boot.configuration.StandaloneConfigura
tionModule@a33c24]
2021-03-16 11:20:12,073 INFO  e.i.e.e.s.i.ExecutionHistoryServiceImpl [main] []
[] - No details history for 6769c266-5080-448f-9104-19c05a0c4adc
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by com.cedarsoftware.util.io.MetaUtils (file:
/home/osmc/work/ibaRPA-0.0.8-full.jar) to field java.lang.Enum.name
WARNING: Please consider reporting this to the maintainers of com.cedarsoftware.
util.io.MetaUtils
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflect
ive access operations
WARNING: All illegal access operations will be denied in a future release
2021-03-16 11:20:12,620 INFO  e.i.e.e.s.i.OneNodeExecutionServiceImpl [main] []
[687db1fe-87e4-4bf9-92bd-c3a6373bf778] - Started class org.example.tasks.RemoteT
ask execution with uuid 687db1fe-87e4-4bf9-92bd-c3a6373bf778
```



```
odule module configuration class candidate for launch.
2021-03-16 11:20:09,390 INFO  e.i.e.engine.boot.ApModuleRunner [main] [] [] - Ad
ded eu.ibagroup.easyrpa.engine.boot.configuration.StandaloneConfigurationModule@
a33c24 into configuration.
2021-03-16 11:20:09,392 INFO  e.i.e.engine.boot.ApModuleRunner [main] [] [] - Fo
und class eu.ibagroup.easyrpa.engine.boot.configuration.NodeConfigurationModule
module configuration class candidate for launch.
2021-03-16 11:20:09,405 INFO  e.i.e.engine.boot.ApModuleRunner [main] [] [] - ea
sy-rpa-common version 1.1.1-24784-06de35f4
2021-03-16 11:20:09,421 INFO  e.i.e.engine.boot.ApModuleRunner [main] [] [] - Ru
nning
module: org.example.ap.Module
uuid: 6769c266-5080-448f-9104-19c05a0c4adc
configuration [eu.ibagroup.easyrpa.engine.boot.configuration.StandaloneConfigura
tionModule@a33c24]
2021-03-16 11:20:12,073 INFO  e.i.e.e.s.i.ExecutionHistoryServiceImpl [main] []
[] - No details history for 6769c266-5080-448f-9104-19c05a0c4adc
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by com.cedarsoftware.util.io.MetaUtils (file:
/home/osmc/work/ibaRPA-0.0.8-full.jar) to field java.lang.Enum.name
WARNING: Please consider reporting this to the maintainers of com.cedarsoftware.
util.io.MetaUtils
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflect
ive access operations
WARNING: All illegal access operations will be denied in a future release
2021-03-16 11:20:12,620 INFO  e.i.e.e.s.i.OneNodeExecutionServiceImpl [main] []
[687db1fe-87e4-4bf9-92bd-c3a6373bf778] - Started class org.example.tasks.RemoteT
ask execution with uuid 687db1fe-87e4-4bf9-92bd-c3a6373bf778
2021-03-16 11:24:28,924 INFO  org.example.tasks.RemoteTask [main] [] [687db1fe-87e4-4bf9-92bd-c3a6373b
f778] - Listing temp location: /tmp
2021-03-16 11:24:28,953 INFO  org.example.tasks.RemoteTask [main] [] [687db1fe-87e4-4bf9-92bd-c3a6373b
f778] - /tmp/systemd-private-e14ca83ba1c74225952ab16524fd9fb3-ntp.service-x2MFkM
2021-03-16 11:24:28,955 INFO  org.example.tasks.RemoteTask [main] [] [687db1fe-87e4-4bf9-92bd-c3a6373b
f778] - /tmp/.font-unix
2021-03-16 11:24:28,957 INFO  org.example.tasks.RemoteTask [main] [] [687db1fe-87e4-4bf9-92bd-c3a6373b
f778] - /tmp/.ICE-unix
2021-03-16 11:24:28,959 INFO  org.example.tasks.RemoteTask [main] [] [687db1fe-87e4-4bf9-92bd-c3a6373b
f778] - /tmp/hsperfdata_root
2021-03-16 11:24:28,962 INFO  org.example.tasks.RemoteTask [main] [] [687db1fe-87e4-4bf9-92bd-c3a6373b
f778] - /tmp/.Test-unix
2021-03-16 11:24:28,964 INFO  org.example.tasks.RemoteTask [main] [] [687db1fe-87e4-4bf9-92bd-c3a6373b
f778] - /tmp/hsperfdata_osmc
2021-03-16 11:24:28,966 INFO  org.example.tasks.RemoteTask [main] [] [687db1fe-87e4-4bf9-92bd-c3a6373b
f778] - /tmp/.X11-unix
2021-03-16 11:24:28,968 INFO  org.example.tasks.RemoteTask [main] [] [687db1fe-87e4-4bf9-92bd-c3a6373b
f778] - /tmp/.XIM-unix
2021-03-16 11:24:30,744 INFO  org.example.tasks.RemoteTask [main] [] [687db1fe-87e4-4bf9-92bd-c3a6373b
f778] - Done
2021-03-16 11:24:30,749 INFO  e.i.e.e.s.i.OneNodeExecutionServiceImpl [main] [] [687db1fe-87e4-4bf9-92
bd-c3a6373bf778] - Finished class org.example.tasks.RemoteTask execution with uuid 687db1fe-87e4-4bf9-
92bd-c3a6373bf778
2021-03-16 11:24:30,765 INFO  e.i.e.engine.boot.ApModuleRunner [main] [] [] - Module run completed wit
h OK
osmc@osmc:~/work$ ls -la
total 129428
drwxr-xr-x  3 osmc osmc      4096 Mar 16 11:20 .
drwxr-xr-x 21 osmc osmc      4096 Mar 15 13:24 ..
-rw-r--r--  1 osmc osmc         0 Mar 16 11:20 elastic.log
-rw-r--r--  1 osmc osmc 132517912 Mar 15 13:27 ibaRPA-0.0.8-full.jar
drwxr-xr-x  3 osmc osmc      4096 Mar 16 11:20 object_storage
osmc@osmc:~/work$
```

Reference links:

Intellij IDEA Tutorial: Remote debug

(v. 2.2) Hybrid Automation

EasyRPA implements an approach widely known as **Hybrid Automation** that allows human employees and RPA robots work together passing tasks back and forth to each other with full visibility. The resulting end-to-end RPA solution combines robot efficiency and scalability with human creativity and flexibility.

Human Task (HT) is a unit of work that involves a human. Quite often, this task requires that the human interact with other services, and thus becomes a task within a larger business goal.

EasyRPA provides automation process developers with a set of (v. 2.2) Pre-defined Human Tasks and framework to (v. 2.2) Set up Human Task.

(v. 2.2) Pre-defined Human Tasks

Human Task Type is a special small Web-application which can read and parse your **Document Type** configuration and it knows how to display input documents.

Document Type is a configuration which lets Human Task know the output fields you want to receive from the document and defines additional information which is used by Human Task to display an input document.

Each **Document Type** relates to some of the **Human Task Type**.

EasyRPA provides 3 predefined Human Task Types:

- (v. 2.2) Information Extraction Human Task
- (v. 2.2) Document Classification Human Task
- (v. 2.2) Forms Human Task

(v. 2.2) Information Extraction Human Task

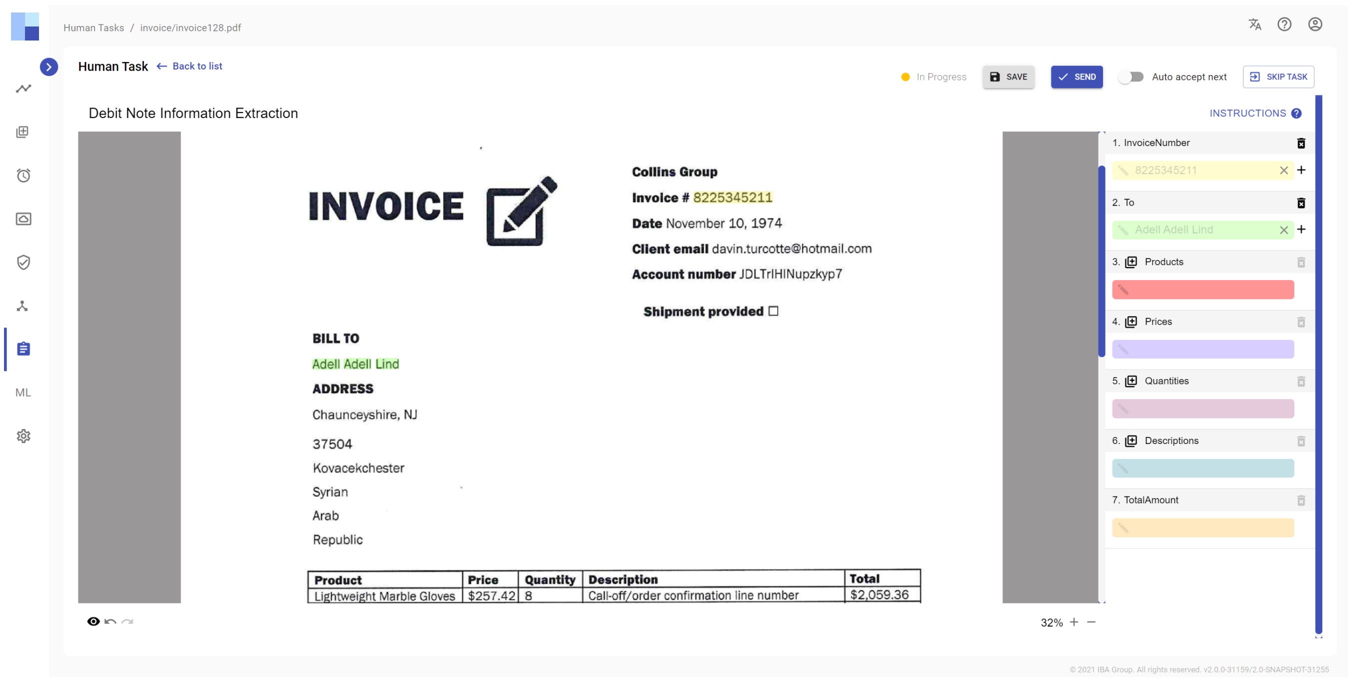
- Document Type JSON Structure
- Hotkeys
- Input JSON Structure
- Output JSON Structure

Information Extraction (IE) is a process of extracting structured information (or key facts) from unstructured and/or semi-structured documents (invoices, claims, dividend news, etc.).

Information Extraction Human Task can be utilized for:

- Saving and processing data extracted by human in automation process
- Collecting Training Set for Machine Learning model training
- Verification of data extracted by Machine Learning model

Example of IE Human Task:



INVOICE

Collins Group
Invoice # 8225345211
Date November 10, 1974
Client email davin.turcotte@hotmail.com
Account number JDLTrlHINupzkyp7

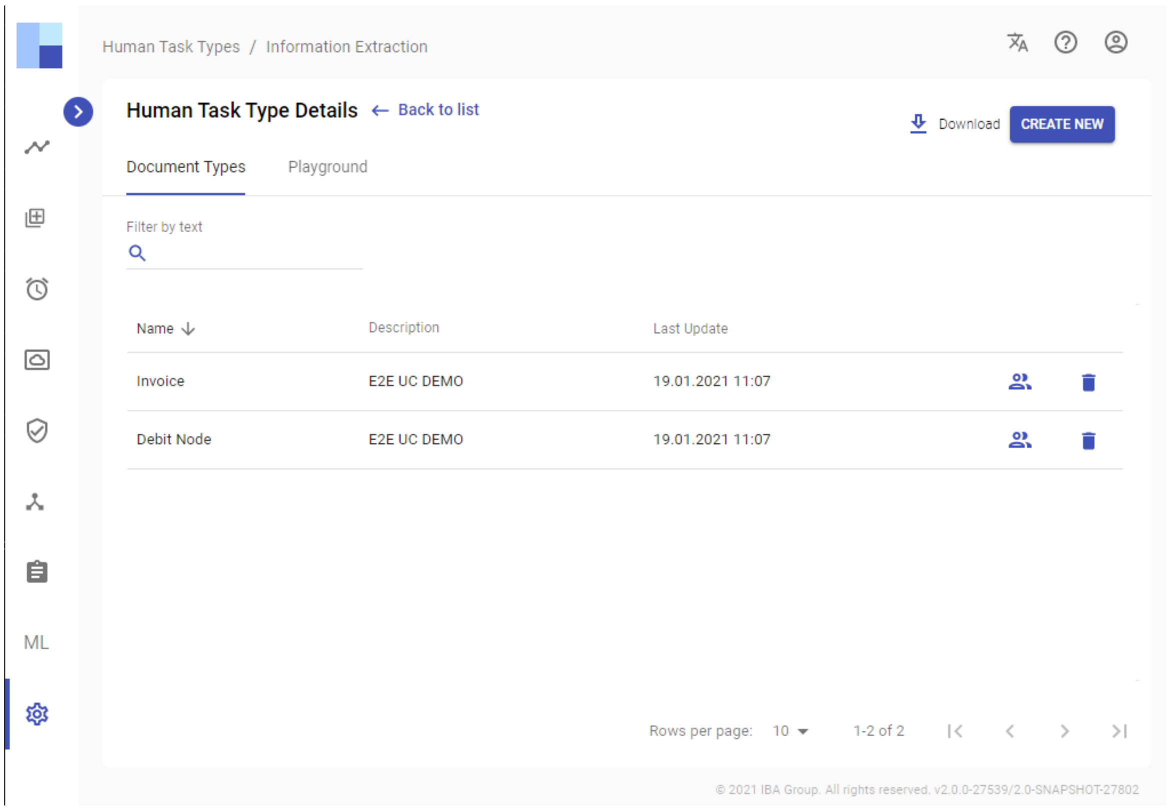
Shipment provided ☐

BILL TO
Adell Adell Lind
ADDRESS
Chaunceyshire, NJ
37504
Kovacekchester
Syrian
Arab
Republic

Product	Price	Quantity	Description	Total
Lightweight Marble Gloves	$257.42	8	Call-off/order confirmation line number	$2,059.36

In order to use the IE Human Task we need to define **Document Type**.

When you open **Information Extraction Human Task Type** there's a list of all document types are related to that Human Task Type, and you will also see "CREATE NEW" button:


Human Task Types / Information Extraction
Back to list
Download
CREATE NEW
Document Types
Playground
Filter by text
ML
Rows per page: 10
1-2 of 2
© 2021 IBA Group. All rights reserved. v2.0.0-27539/2.0-SNAPSHOT-27802

Human Task Type Details

Name	Description	Last Update
Invoice	E2E UC DEMO	19.01.2021 11:07
Debit Node	E2E UC DEMO	19.01.2021 11:07

You need to provide the **name**, **description** and **settings** for new Document Type.

Document Type JSON Structure

Settings is a configuration in JSON format which has the structure as in example bellow:

Settings example

```
{
         "regexFunctions": {
                "numberMore10": "(value) => { return Number(value) > 10}",
                "lengthMore3": "(value) => { return value.length > 3}"
        },
    "taskInstructionText": "Carefully read and classify this text using the categories provided.",
    "taskInstructionLink": "https://172.20.194.57:8444/minio/data/ht_articles_classification/instruction.html",
    "taskTypeLabel": "Information Extraction",
    "autoSave": false,
    "allowCustomValue": false,
    "appLanguage": "en",
    "excludeUndefinedEntities": false,
        "colors": [
                "000000",
                "#ffffff"
        ],
        "categories": [
                {
                        "name": "Amount",
                        "multiple": true,
                        "regex": "^[0-9]*$",
                        "hotkey": ["ctrl", "q"]
                },
                {
                        "name": "Description",
                        "multiple": true,
                         "hotkey": ["G"]
                },
                {
                        "name": "ItemQuantity",
                        "multiple": true,
                        "required": true,
                         "hotkey": ["Ctrl", "Shift", "E"]
                },
                {
                        "name": "ItemPrice",
                        "multiple": true,
                        "required": true,
                        "helperText": "Number should be more than 10",
                        "regexFnc": ["numberMore10","lengthMore3"]
                },
                {
                        "name": "ItemAmount",
                        "multiple": true
                }
        ]
}
```

This settings contains:

- **regexFunctions** (object) (optional) - special object which contains "key" "value" pairs to specify custom fields validators, where "key" is your custom function name which can be used as a link in "**regexFnc**" option of field settings under "**categories**" setting. "value" - is your custom JavaScript lambda function.
- **taskInstructionText** (string) (optional) - represents the instructions text that will appear in the popup
- **taskInstructionLink** (string) (optional) - represents the link to the remote instructions source. Having at least one of these fields **taskInstructionText** or **taskInstructionLink** will cause the Instructions button to appear.
- **taskTypeLabel** (string) (optional) - allows to configure the task title. By default it is set to "Information Extraction".
- **autoSave** (boolean) (optional) - allows to save intermediate task results automatically. Has ***false*** by default.
- **allowCustomValue** (boolean) (optional) - enables the possibility to enter a custom value without any connection to the input document picture. This possibility is useful when OCR failed to extract some text from the document.
- **appLanguage** (string) (optional) - allows user to setup the Human Task localization. Currently available options are "*en*" and "*ru*
". By default the task displays as "*en*".
- **excludeUndefinedEntities** (boolean) (optional) - allows to get the output only of fields which are configured in "categories" setting. By default the setting has "*false*" value.
- **colors** (list of string) (optional) - list of hex colors (format: 6-character string with an optional # at the beginning), that will be used before the predefined HTT colors and applied to category elements.
- **categories** (list of objects) (required) - list of fields to extract from the document, where each item contains:
  - **name** (string) (required) - it serves as a key to receive data from Human Task output
  - **multiple** (boolean) (required) - shows if this field may have multiple values (uses when you have the list with details of some items in your document, e.g. list of products)
  - **required** (boolean) (optional) - shows if field is required for extraction, so human won't be able to submit Human Task without specifying the values for all required fields
  - **helperText** (string) (optional) - additional text which is shown if value is invalid due to additional validators specified by regexp settings.
  - **regex** (string) (optional) - specify regexp function to use for values validation
  - **regexFnc** (list of string) (optional) - list of names of validator functions, which are specified by in "**regexFunctions**".
  - **hotkey** (list of string) (optional) - list of keyboard shortcuts that can be pressed to selection of the last item in the category.

Category hotkey rules:

- should not be busy yet;
- should not be repeated;
- should not be contain multiple letters or digits;
- should contain only existing keyboard keys;
- can be in any register;
- can be one or more keys.

Hotkeys

Information Extraction Human Task has next hotkeys:

- **Ctrl + Z** - cancel last action;
- **Ctrl + Shift + Z** - return the last canceled action back;
- **Shift + Tab** - selecting the previous category field;
- **Tab** - selecting the next category field;
- **Ctrl + Mouse Wheel** - zoom in or zoom out of the image;
- **Shift + Area Selection** on image - attaching of selected words to the end of the current category field value (appending of words);
- (Only for **multiple** categories) **Shift + Plus Sign (+)** - add an empty field to current category;
- **Enter** - enter/exit edit mode of the current category field;
- (Only edit mode) **Shift + Enter** - add a line break character;
- **Esc** - cancel appending mode for current category field, in edit mode - undo changes and exit edit mode.

Input JSON Structure

The input for IE Human Task is a JSON with the following structure:

input Json example

```
{
	"images": [ // There should be an array to support multipage PDFs. Each page should be defined
individually in this case.
		{
			"content": "https://some-provider.com/img.jpg",
			"json": <OCR-JSON>,
			"dimensions": { // The dimensions of the page in pixels.
				"width": 2000,
				"height": 3000
			}
		}
	]
}
```

Input JSON contains:

- **images** (list of objects) - the root element which contains the list of document configurations. In common cases there's always 1 element in the list.
    - **content** (url or base64 string) - source of the input document to display. It may be an **URL** to the document or document content encoded in **base64** format (e.g. the string value "data:image/jpg;base64,R0lGOD....").
    - **json** (**OCR-JSON** object) - provides OCR information. OCR-JSON structure is described bellow.
    - **dimensions** (object) - an object contains "width" and "height" parameters which represent width and height of original input document.
        - **width** (integer) - width value of original input document.
        - **height** (integer) - height value of original input document.

OCR-JSON object has the following structure which is generated by OCR component by itself:

OCR-JSON

```
"json": {
	"pages": [
		"id": "page0",
		"areas": [
			{
				"id": "page0_area0",
				"paragraphs": [
					{
						"id": "page0_area0_paragraph0",
						"lines": [
							{
								"id": "page0_area0_paragraph0_line0",
								"words": [
									{
										"id":
```

```
"page0_area0_paragraph0_line0_word0",
                                                                                    "text": "Advanced",
                                                                                    "properties": {
                                                                                            "bbox": [

0.05999032414126754,

0.037290455011974,

0.14078374455732948,

0.046527540198426275
                                                                                            ],
                                                                                    "x_fsize": 0,
                                                                                    "x_wconf": 96
                                                                            }
                                                                    },
                                                            }
                                                    ...
                                                    ]
                                            }
                                    ]
                            }
                    ]
            }
```

This JSON contains all OCRed words and kept in tree-like structure: **pages areas paragraphs lines words**

- **json** (object) - root element
    - **pages** (list of object) - list of pages structure. Each page in the list has the following structure:
        - **id** (string) - id of the page
        - **areas** (list of object) - list of areas structure. Each area in the list has the following structure:
            - **id** (string) - id of the area
            - **paragraphs** (list of object) - list of paragraphs structure. Each paragraph in the list has the following structure:
                - **id** (string) - id of the paragraph
                - **lines** (list of object) - list of lines structure. Each line in the list has the following structure:
                    - **id** (string) - id of the line
                    - **words** (list of object) - list of words structure. Each word in the list has the following structure:
                        - **id** (string) - id of the word
                        - **text** (string) - original text extracted by OCR engine
                        - **properties** (object) - property object which the the following structure:
                            - **bbox** (list of integers) - top-left and bottom-right coordinates of the rectangle around the word in original document. Coordinates are normalized to be from 0 to 1 relative to original document size
                        - **x_fsize** (integer) - is the OCR-engine specific font size
                        - **x_wconf** (integer) - OCR-engine specific confidence for the entire contained substring. Higher values expresses higher confidences

Output JSON Structure

As an output IE Human Task produces the following JSON:

Output JSON example

```
{
        "entities": [
                {
                        "content": "€2,000.00",
                        "name": "Amount",
                        "words": [
                                {
                                        "content": "€2,000.00",
                                        "bbox": [
                                                0.8490566037735849,
                                                0.1091344509066028,
                                                0.9395258829221094,
                                                0.12145056448853918
                                        ],
                                        "id": "page0_area2_paragraph1_line0_word6",
                                        "page": 0
                                }
                        ],
                        "index": 0,
                        "multiple": true
                }
        ]
}
```

It has the following structure:

- **entities** (list of objects) - root element which contains a list of extracted entities. Each entity in this list has the structure as described below:
    - **content** (string) - final output text of the extracted entity.
    - **name** (string) - extracted entity name.
    - **words** (list of objects) - list of word objects. If extracted text has several words, this list will contain several word object with the following structure:
        - **content** (string) - original text from the input document
        - **bbox** (list of integers) - top-left and bottom-right coordinates of the rectangle around the word in original document. Coordinates are normalized to be from 0 to 1 relative to original document size
        - **id** (string) - id of the word
        - **page** (integer) - page number where word is placed in original document
    - **index** (integer) - entity index
    - **multiple** (boolean) - shows if this field may have multiple values

(v. 2.2) Document Classification Human Task

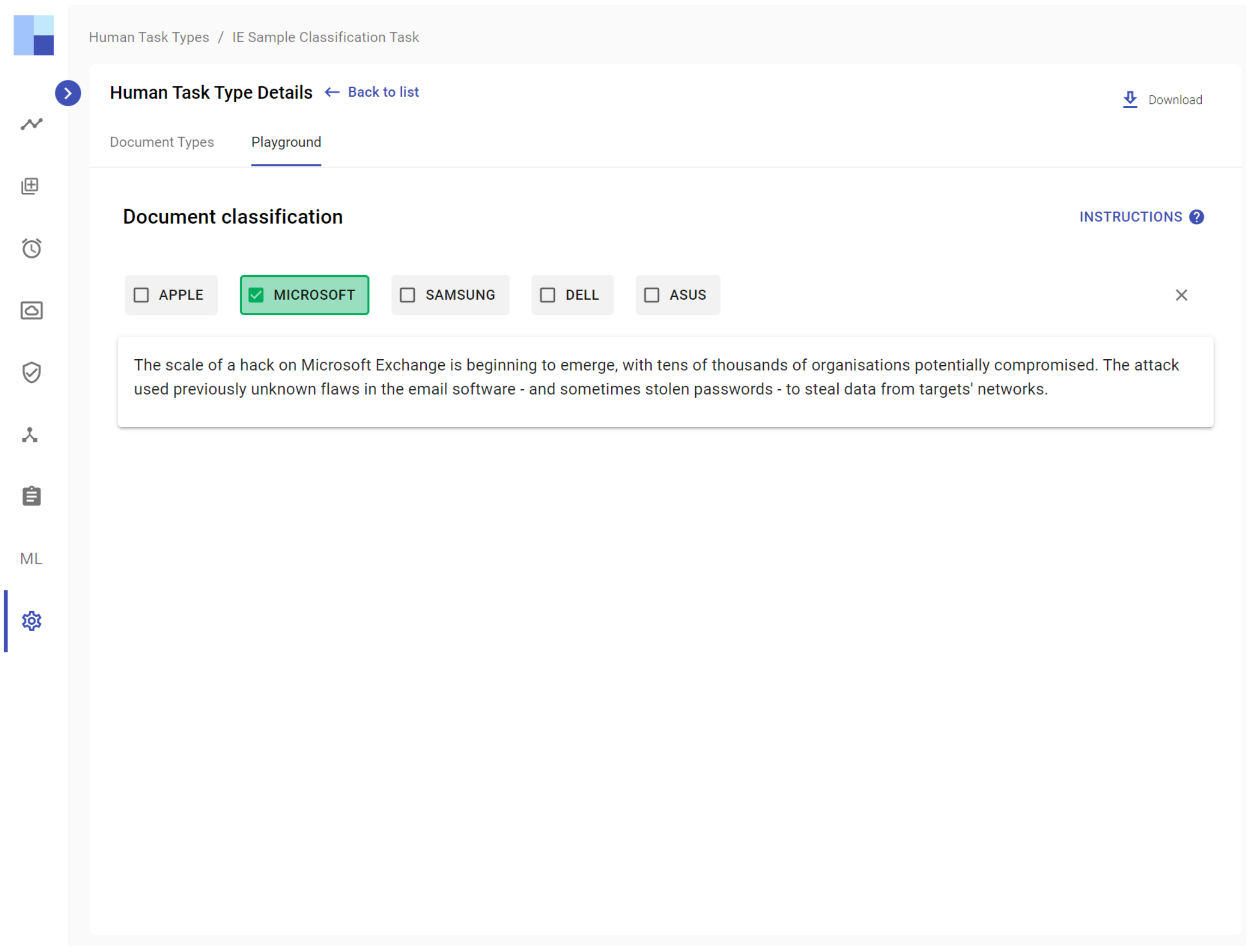
Document Classification Human Task is used to assign a document to one or more classes or categories.

Document Classification Human Task can be configured for **Single Label Classification** (when 1 document may relate only to one category) or **Multi Label Classification** (when 1 document may relate to multiple categories)

Document Classification Human Task can be utilized for:

- Saving and processing category classified by human in automation process
- Collecting Training Set for Machine Learning model training
- Verification of data extracted by Machine Learning model

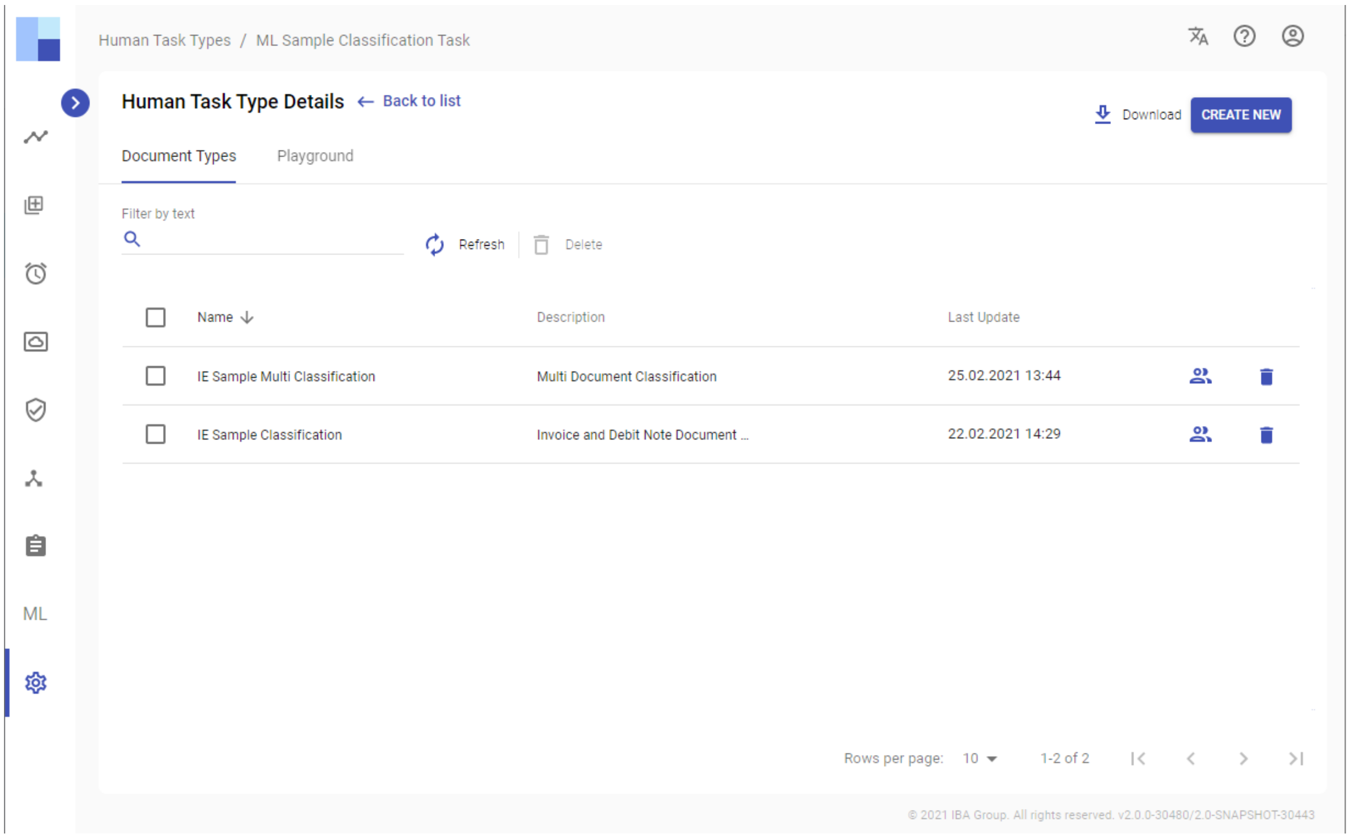
Example of Classification Human Task:



The scale of a hack on Microsoft Exchange is beginning to emerge, with tens of thousands of organisations potentially compromised. The attack used previously unknown flaws in the email software - and sometimes stolen passwords - to steal data from targets' networks.

In order to use the Classification Human Task we need to define **Document Type**.

When you open **Document Classification Human Task Type** there's a list of all document types are related to that Human Task Type, and you will also see "CREATE NEW" button:



Name	Description	Last Update
IE Sample Multi Classification	Multi Document Classification	25.02.2021 13:44
IE Sample Classification	Invoice and Debit Note Document ...	22.02.2021 14:29

Rows per page: 10
1-2 of 2
© 2021 IBA Group. All rights reserved. v2.0.0-30480/2.0-SNAPSHOT-30443

You need to provide the **name**, **description** and **settings** for new Document Type.

Document Type JSON Structure

Settings is a configuration in JSON format which has the structure as in example bellow:

Document Type JSON example

```
{
    "autoSave": false,
    "taskInstructionText": "Carefully read and classify this text using the categories provided.",
    "taskInstructionLink": "https://172.20.194.57:8444/minio/data/ht_articles_classification/instruction.html",
    "taskTypeLabel": "Document Classification",
    "appLanguage": "en",
        "multipleChoice": false,
        "categories": [
                "Science",
                "Politics",
                "Sport",
                "Business"
        ],
        "scoreThreshold": 0.1
}
```

This settings contains:

- **autoSave** (boolean) (optional) - allows to save intermediate task results automatically. Has ***false*** by default.
- **taskInstructionText** (string) (optional) - represents the instructions text that will appear in the popup
- **taskInstructionLink** (string) (optional) - represents the link to the remote instructions source. Having at least one of these fields **taskInstructionText** or **taskInstructionLink** will cause the Instructions button to appear.
- **taskTypeLabel** (string) (optional) - allows to configure the task title. By default it is set to "Document Classification".
- **appLanguage** (string) (optional) - allows user to setup the Human Task localization. Currently available options are "*en*" and "*ru*". By default the task displays as "*en*".

- **multipleChoice** (boolean) (optional) - allows to enable multiple choice mode, where you can select multiple categories instead of one. By default the setting has "*false*" value.
- **categories** (list of strings) (required) - list of predefined classes your documents may relate to
- **scoreThreshold** (decimal) (optional) - is used when **multipleChoice is true** to auto select the categories after ML. When **multipleChoice is false**, the **scoreThreshold** doesn't matter because the category with highest score is automatically selected.

Input JSON Structure

The input for Document Classification Human Task is a JSON with the following structure:

Input JSON example

```
{
        "text": [
                "Lorem Ipsum is simply dummy text of the printing and typesetting industry.",
                "Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown
printer took a galley of type and scrambled it to make a type specimen book.",
                "It has survived not only five centuries, but also the leap into electronic typesetting,
remaining essentially unchanged.",
                "It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum
passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem
Ipsum."
        ]
}
```

Input JSON contains:

- **text** (list of strings) - list of texts to classify. In common case there's only 1 string inside the list. If we have multiple texts in the list - it will be merged into the single one on Human Task user interface.

Output JSON Structure

As an output Document Classification Human Task produces the following JSON:

outputJson example

```
{
        "categories": [
                "Science"
        ],
        "scores": {
                "Science": 0.7745,
                "Politics": 0.211,
                "Sport": 0.06,
        }
}
```

It has the following structure:

- **categories** (list of strings) - the list of result categories input document relates to. This result can be provided either by ML or by a human. Can have multiple values in the list if Document Type setting has **multipleChoice is true,** and only 1 value in another case.
- **scores** (object) - it's an optional field which exist only if classification results provided by ML. It shows the score value for each category as a **key value** structure**,** where is the **key** - is category name and **value** is decimal value represents a score from 0 to 1.

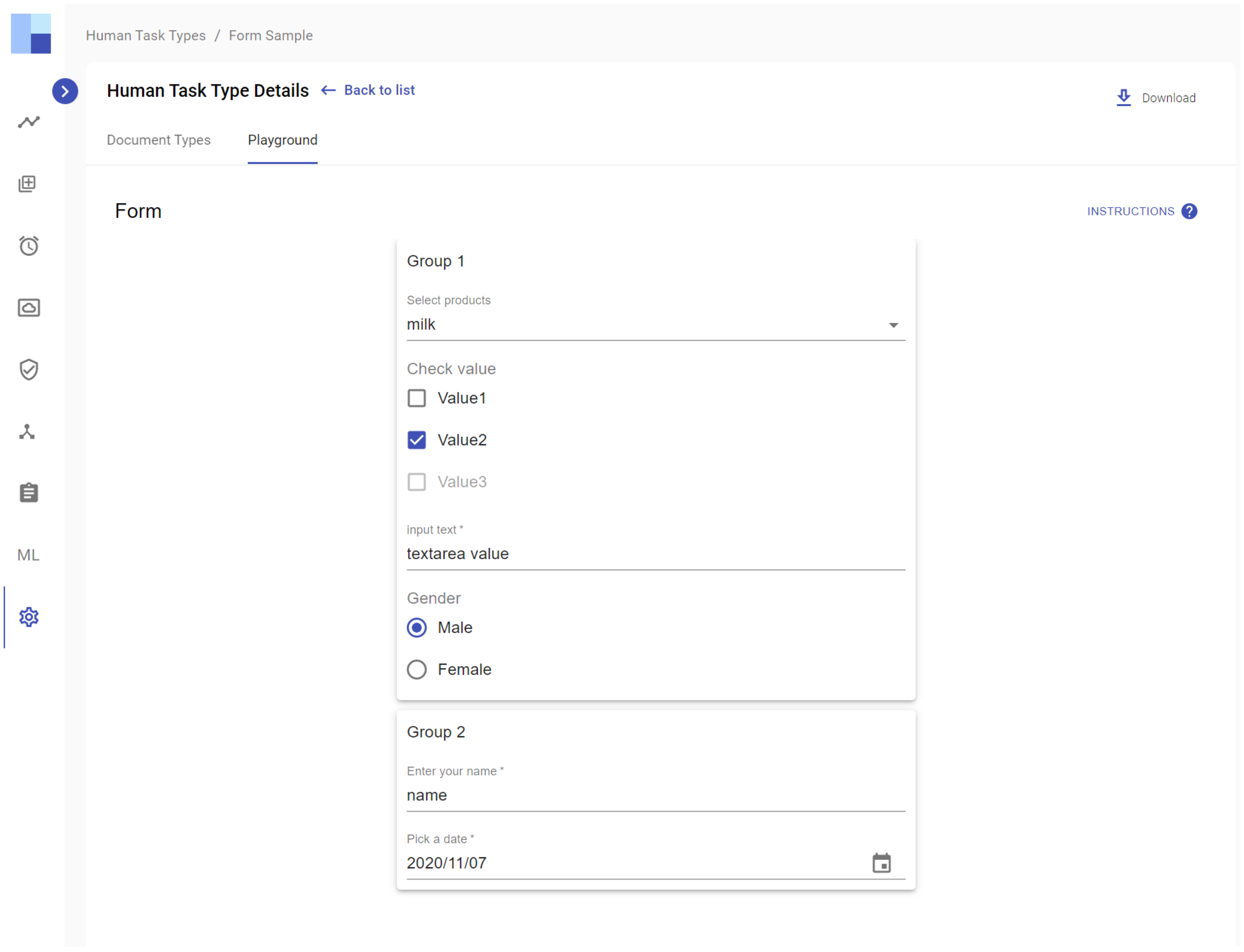
(v. 2.2) Forms Human Task

Forms Human Task provides possibility to setup a Human Task which contains several form elements, in order to perform some sort of surveys. The task supports the next set of fields:

- Simple text (single line input field) (type **text**);
- Multiline text (multiline input field) (type **textarea**);
- Date picker (type **date**);
- Radio buttons (elements of single choice) (type **radio**);
- Select dropdown lists (single choice from the list of values, better if you have long list to choose from) (type **select**);
- Checkbox for multiple choice (multiple choices from the list of values) (type **checkbox**).

The question list might be divided into separate blocks using groups. They are setup in the settings as an array of input types.

Example of Forms Human Task:



Form

Group 1

Select products
milk

Check value
- [ ] Value1
- [x] Value2
- [ ] Value3

input text *
textarea value

Gender
[✓ Male] Female

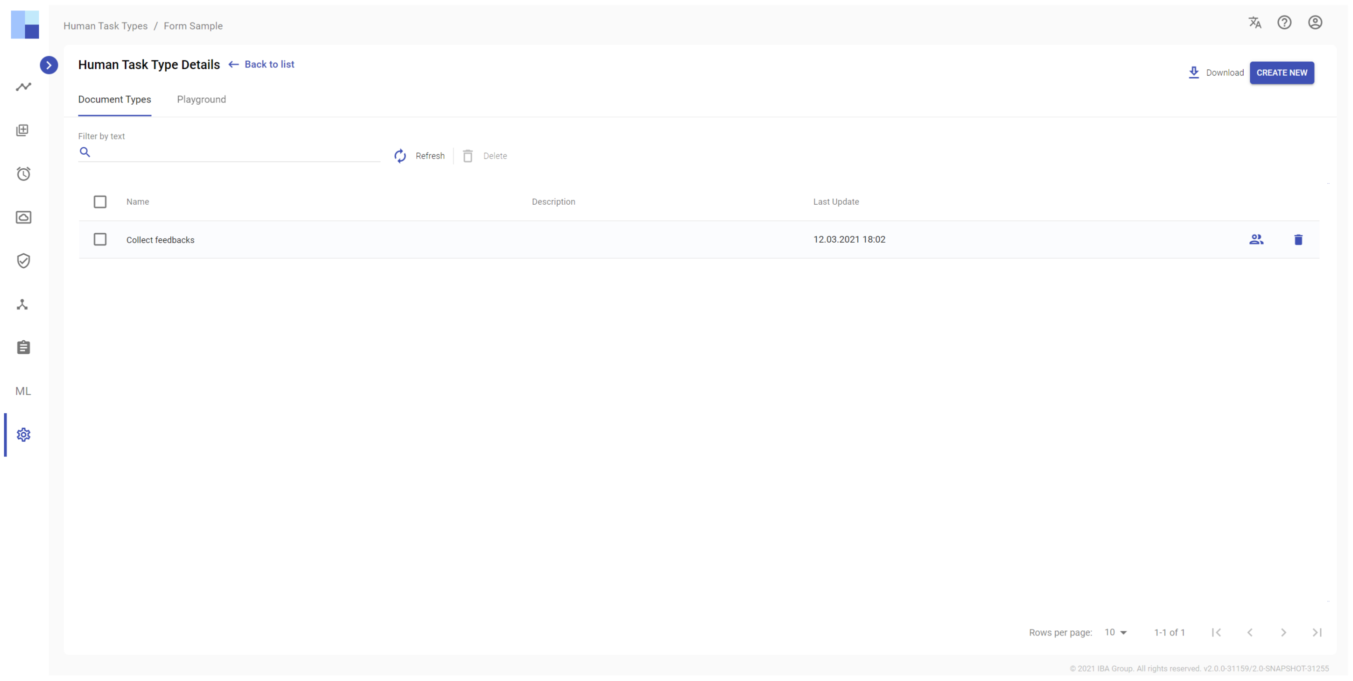
Group 2

Enter your name *
name

Pick a date *
2020/11/07

In order to create a forms in Forms Human Task we need to define Document Type.

When you open **Forms Human Task Type** there's a list of all document types are related to that Human Task Type, and you will also see "CREATE NEW" button:


Human Task Types / Form Sample

Human Task Type Details ← Back to list

Download CREATE NEW

Document Types Playground

Filter by text

Refresh Delete

	Name	Description	Last Update
☐	Collect feedbacks		12.03.2021 18:02

Rows per page: 10 1-1 of 1

© 2021 IBA Group. All rights reserved. v2.0.0-31159/2.0-SNAPSHOT-31255

You need to provide the **name**, **description** and **settings** for new Document Type.

Document Type JSON Structure

Here is an example of settings object that defines the form:

Forms Settings JSON example

```
{
    "autoSave": false,
    "taskInstructionText": "Carefully read and classify this text using the categories provided.",
    "taskInstructionLink": "https://172.20.194.57:8444/minio/data/ht_articles_classification/instruction.html",
    "taskTypeLabel": "Forms",
    "appLanguage": "en",
        "groups": [
                {
                        "groupTitle": "General Information",
                        "fields": [
                                {
                                        "name": "full_name",
                                        "label": "Please, provide your Full Name",
                                        "type": "text",
                                        "required": true,
                                        "validationRegExp": "/^[a-zA-Z]+$/"
                                },
                                {
                                        "name": "email",
                                        "label": "Please, provide your email",
                                        "type": "text",
                                        "required": true,
                                        "validationRegExp": "^[\\w-\\.]+@([\\w-]+\\.)+[\\w-]{2,4}$"
                                },
                                {
                                        "name": "birthday",
                                        "label": "Please, provide your birthday",
                                        "type": "date",
                                        "required": true
                                }, {
                                        "name": "Gender",
                                        "label": "Please, provide your gender",
                                        "type": "radio",
                                        "required": false,
                                        "items": [
                                                { "label": "Male", "disabled": false },
                                                { "label": "Female", "disabled": false },
                                                { "label": "Other", "disabled": true }
                                        ]
                                }
                        ]
                },
                {
                        "groupTitle": "Provide your feedback",
                        "fields": [
                                {
                                        "name": "goal_of_using",
                                        "label": "What is your main goal for using this website?",
                                        "type": "textarea",
                                        "required": true
                                },
                                {
                                        "name": "what_changed",
                                        "label": "What changed for you after you started using our website?",
                                        "type": "textarea",
                                        "required": true,
                                        "validationRegExp": "/^[a-zA-Z]{20,50}$/"
                                },
                                {
                                        "name": "where_did_you_hear",
                                        "label": "Where did you first hear about us?",
                                        "type": "radio",
                                        "required": true,
                                        "items": [
                                                {
                                                        "label": "On Google",
                                                        "disabled": false
                                                },
                                                {
                                                        "label": "On Billboards",
                                                        "disabled": false
                                                },
                                                {
                                                        "label": "On TV Ads",
                                                        "disabled": false
                                                }
                                        ]
                                },
                                {
                                        "name": "what_you_like",
                                        "label": "What do you like most about our website?",
                                        "type": "select",
                                        "required": true,
                                        "values": ["Design", "Experience", "Action Flow", "Ease of finding
required information"]
                                },
                                {
                                        "name": "what_you_interested",
                                        "label": "what are you interested in?",
                                        "type": "checkbox",
                                        "required": true,
                                        "disabled": false,
                                        "items": [
                                                { "label": "Politic", "disabled": false},
                                                { "label": "Sport", "disabled": false},
                                                { "label": "History", "disabled": true }
                                        ]
                                }
                        ]
                }
        ]
}
```

This settings contains:

- **autoSave** (boolean) (optional) - allows to save intermediate task results automatically. Has ***false*** by default.
- **taskInstructionText** (string) (optional) - represents the instructions text that will appear in the popup
- **taskInstructionLink** (string) (optional) - represents the link to the remote instructions source. Having at least one of these fields **taskInstructionText** or **taskInstructionLink** will cause the Instructions button to appear.
- **taskTypeLabel** (string) (optional) - allows to configure the task title. By default it is set to "Forms".
- **appLanguage** (string) (optional) - allows user to setup the Human Task localization. Currently available options are "*en*" and "*ru*". By default the task displays as "*en*".
- **groups** (list of objects) (required) - list of groups settings. Each group in the list has the following structure:
    - **groupTitle** (string) (required) - identifies the group name to display in Human Task.
    - **fields** (list of objects) (required) - list of fields settings. Each field in the list has the following structure:
        - **name** (string) (required) - is used as a key to get value from output result. It's never displayed on the Human Task and **must be unique for the entire form**
        - **label** (string) (required) - label to display on Human Task.
        - **type** (string) (required) - specifies the type of field. Should be one of the supported types: **text**, **textarea**, **date**, **radio**, **select**, **checkbox**.
        - **required** (boolean) (optional) - shows if field is required for filling, so human won't be able to submit Human Task without specifying the values for all required fields.
        - **validationRegExp** (string) (optional) - provides possibility to validate filled values using regular expressions.
        - **disabled** (boolean) (optional) - provides possibility to disable the field which will prevent user from filling this field.
        - **values** (list of strings) (required only for "type": "select") - is used only if **"type": "select"**. Specifies possible items in the select dropdown list.
        - **items** (list of objects) (required only for "type": "checkbox") - is used only if **"type": "checkbox"**. Specifies possible items to check. Each item in the list has the following properties:
            - **label** (string) (required) - label of the checkbox to display on Human Task.
            - **disabled** (boolean) (optional) - provides possibility to disable the checkbox item which will prevent user from select it.

Input JSON Structure

Below is an example of Input JSON:

Input JSON Example

```
{
        "EmailTextTypeField": "mail",
        "SelectTypeField": "milk",
        "NewYearDateTypeField": "2020-11-12",
        "Checkbox1": [
                "Value1",
                "Value2"
        ],
        "radioButtons": "Male",
        "Name": "name",
        "Birthday": "2020-11-07",
        "textarea1": "textarea value"
}
```

Here we defined the **key value** pairs, where **key** is the unique name of your field on form and **value** - is the value to put into that field.

By providing and input we can pre-populate some fields on form. For example, by JSON above fields with names **EmailTextTypeField, SelectTypeField** and **NewYearDateTypeFieldwill** will be populated with values *"mail", "milk",* and *"2020-11-12"* accordingly.

Output JSON Structure

Output JSON is completely the same as an Input Structure with the **key value** pairs, where **key** is the unique name of your field on form and **value** - is the result value of that field.

Output JSON Example

```
{
        "EmailTextTypeField": "mail",
        "SelectTypeField": "milk",
        "NewYearDateTypeField": "2020-11-12",
        "Checkbox1": [
                "Value1",
                "Value2"
        ],
        "radioButtons": "Male",
        "Name": "name",
        "Birthday": "2020-11-07",
        "textarea1": "textarea value"
}
```

(v. 2.2) Set up Human Task

(v. 2.2) Create Human Task Type

What is Human Task Type ?

A **Human Task Type (HTT)** is a web-based application and part of **EasyRPA** which provides an user interface to interact with Workers (contractors, SMEs, employees) and **Automation Process**.

Basic Structure

Basic **HTT** consists of 3 files:

- index.html - Provides the main logic and html structure. This file is the point of entry in **Human Task**. The basic example of structure bellow:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Human Task Example</title>
</head>
<body>
    <div id="root">
                <!-- some html structure -->
    </div>
</body>
<script>
        //scripts with some rules, logic and etc.
</script>
</html>
```

- sample.json - this file is needed for the Playground where you can see and test your **HTT**.

> The properties *'inputJson', 'outputJson' and 'settings'* are required

```
{
  "inputJson": {
        ...
  },
  "outputJson": {
                ...
  },
  "settings": {
    ...
  }
}
```

- package.json - this file is needed to know the name and version of **HTT** for **EasyRPA**. The basic example bellow:

```
{
  "name": "human-task-type-example",
  "version": "0.1.0"
}
```

Workspace Integration

Workspace uses posting\listening messages into\from **Human Task** to send and receive the data. The **`window.postMessage()`** method safely enables cross-origin communication between `Window` objects. When **Human Task** is created on **Workspace** you need to receive the input json data from **Application Process**. The input data will be stored in '*data*' property of the received message. The structure of input data is the same as *sample.json* which described above.

To get input data you need to add the following listener in your *index.html* or your script file.
'validation-message' means that Worker has completed manual task and it is ready to be completed. Here you can validate the inserted data and post it to **Control Server** and complete **Human Task**.

```
window.addEventListener('message', message => {
    const { data } = message; //collect input data
    if (!data['validation-message']) {
        renderApp(data);
    } else if (data['validation-message']) {
        submit(data);
    }
});
```

When you got the input data you can use *'inputJson'* to fill some fields or use *'settings'* to collect some internal settings eg: labels, regexp, list of required fields etc.
Bellow the example to use incoming settings and inputJson to render Human Task and fill the values in the input field.

```
function renderApp(data) {
        const template = document.createElement('div');
        const root = document.getElementById('root');
        root.appendChild(template);

        const input = document.getElementById('main-input');
        input.value = data.inputJson.inputText;

        const inputLabel = document.getElementById('main-label');
        inputLabel.innerHTML = data.settings.input.label;
}
```

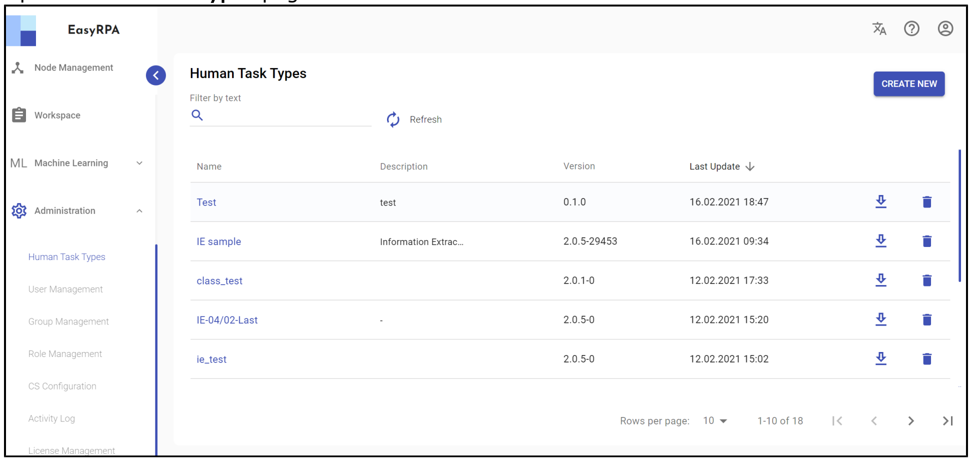
To submit the output data use the next example:

```
function submit() {
        const inputValue = document.getElementById('main-input').value;
        const outputJson = { text: inputValue };
        window.parent.postMessage(outputJson, '*');

        const data = {
                'validation-message': true,
                result: true,
        };
        window.parent.postMessage(data, '*');
}
```

Deploy in Control Server

1. To deploy HTT to the Control Server you need to build a zip archive containing all the application files.

2. Open **Human Task Types** page on **Control Server**

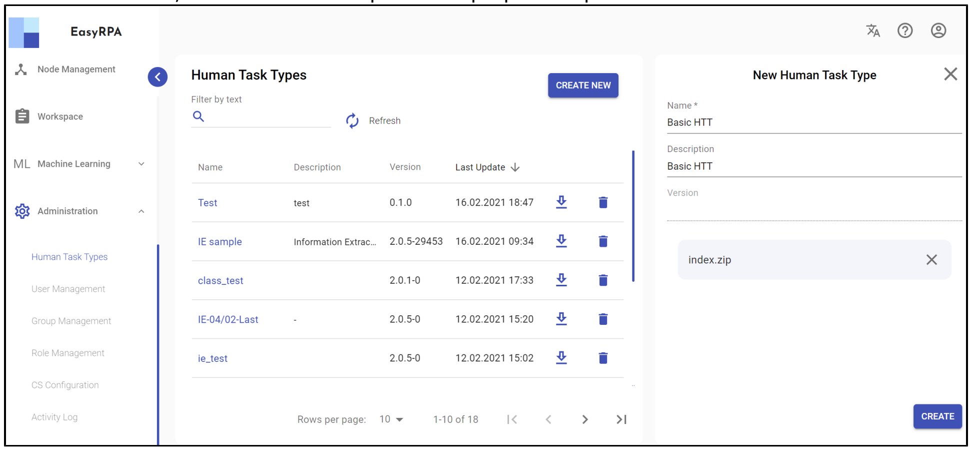


Human Task Types

Name	Description	Version	Last Update
Test	test	0.1.0	16.02.2021 18:47
IE sample	Information Extrac...	2.0.5-29453	16.02.2021 09:34
class_test		2.0.1-0	12.02.2021 17:33
IE-04/02-Last	-	2.0.5-0	12.02.2021 15:20
ie_test		2.0.5-0	12.02.2021 15:02

Rows per page: 10 1-10 of 18

3. Click **Create New**, fill the form and upload the prepared zip archive.



Human Task Types

Name	Description	Version	Last Update
Test	test	0.1.0	16.02.2021 18:47
IE sample	Information Extrac...	2.0.5-29453	16.02.2021 09:34
class_test		2.0.1-0	12.02.2021 17:33
IE-04/02-Last	-	2.0.5-0	12.02.2021 15:20
ie_test		2.0.5-0	12.02.2021 15:02

Rows per page: 10 1-10 of 18

New Human Task Type

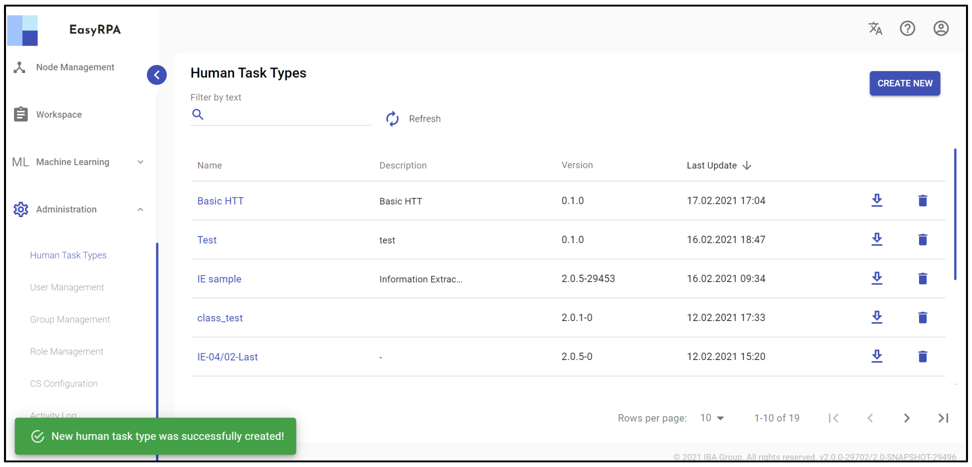
Name *
Basic HTT

Description
Basic HTT

Version

index.zip

4. Click **Create**.



Human Task Types

Name	Description	Version	Last Update
Basic HTT	Basic HTT	0.1.0	17.02.2021 17:04
Test	test	0.1.0	16.02.2021 18:47
IE sample	Information Extrac...	2.0.5-29453	16.02.2021 09:34
class_test		2.0.1-0	12.02.2021 17:33
IE-04/02-Last	-	2.0.5-0	12.02.2021 15:20

Rows per page: 10 1-10 of 19

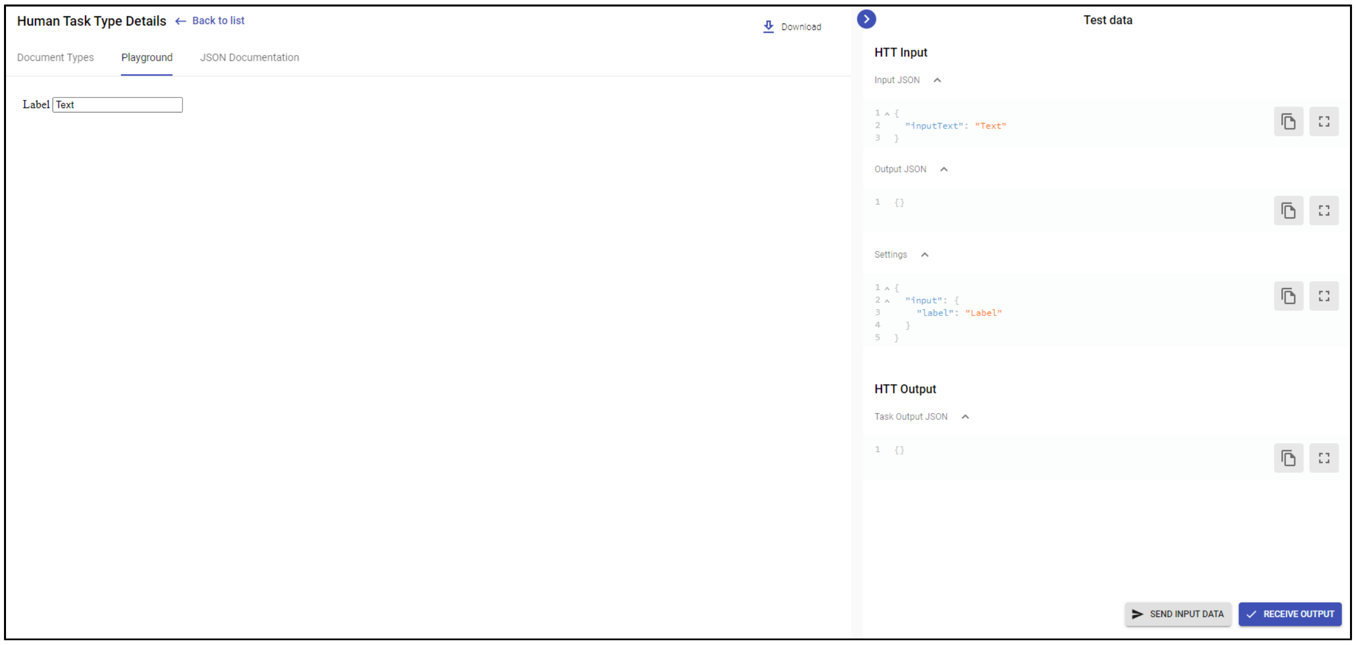
New human task type was successfully created!

© 2021 IBA Group. All rights reserved. v2.0.0-29702/2.0-SNAPSHOT-29496

Test your own Human Task Type

The Playground allows developer to test how the Human Task integrates with the Workspace.

Lets see how the created HTT looks like in the Playground.



Label Text

Input JSON:

```
{
   "inputText": "Text"
}
```

Output JSON:

```
{}
```

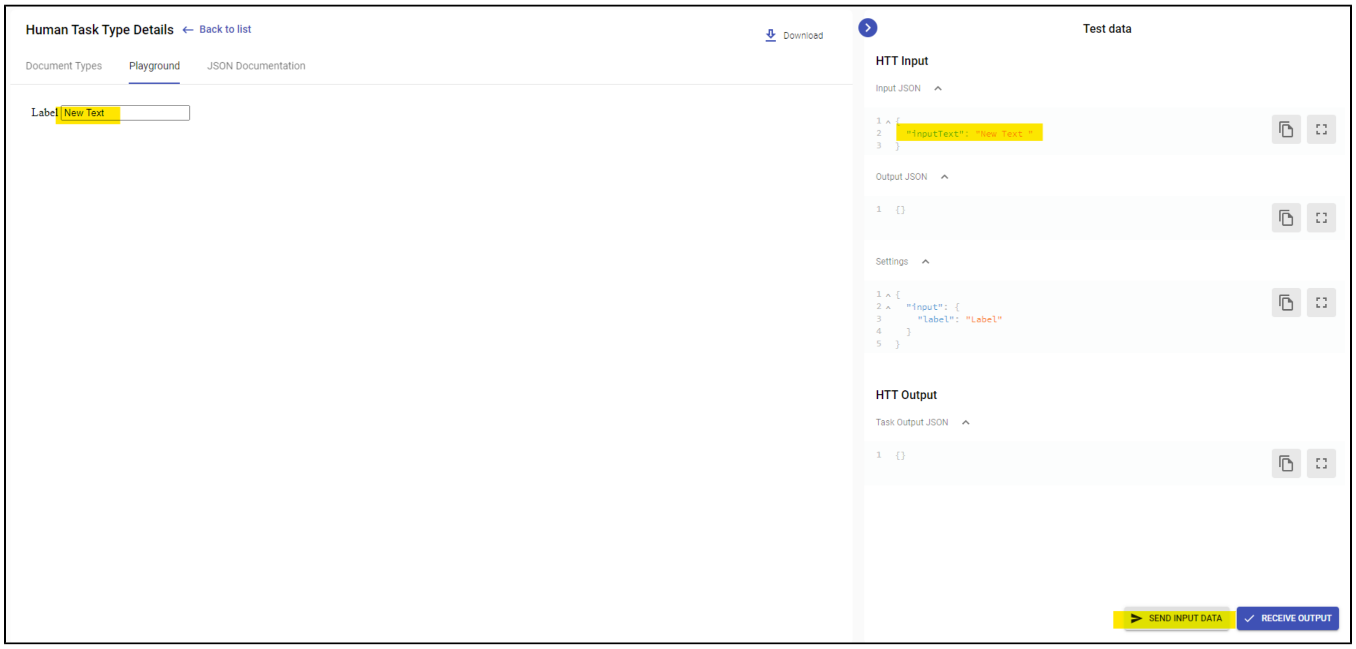
Settings:

```
{
   "input": {
     "label": "Label"
   }
}
```

Task Output JSON:

```
{}
```

You can change the "Input JSON" and click "SEND INPUT DATA". The Workspace will fire an event with new input for the HTT.



Label New Text

Input JSON:

```
{
   "inputText": "New Text "
}
```

Output JSON:

```
{}
```

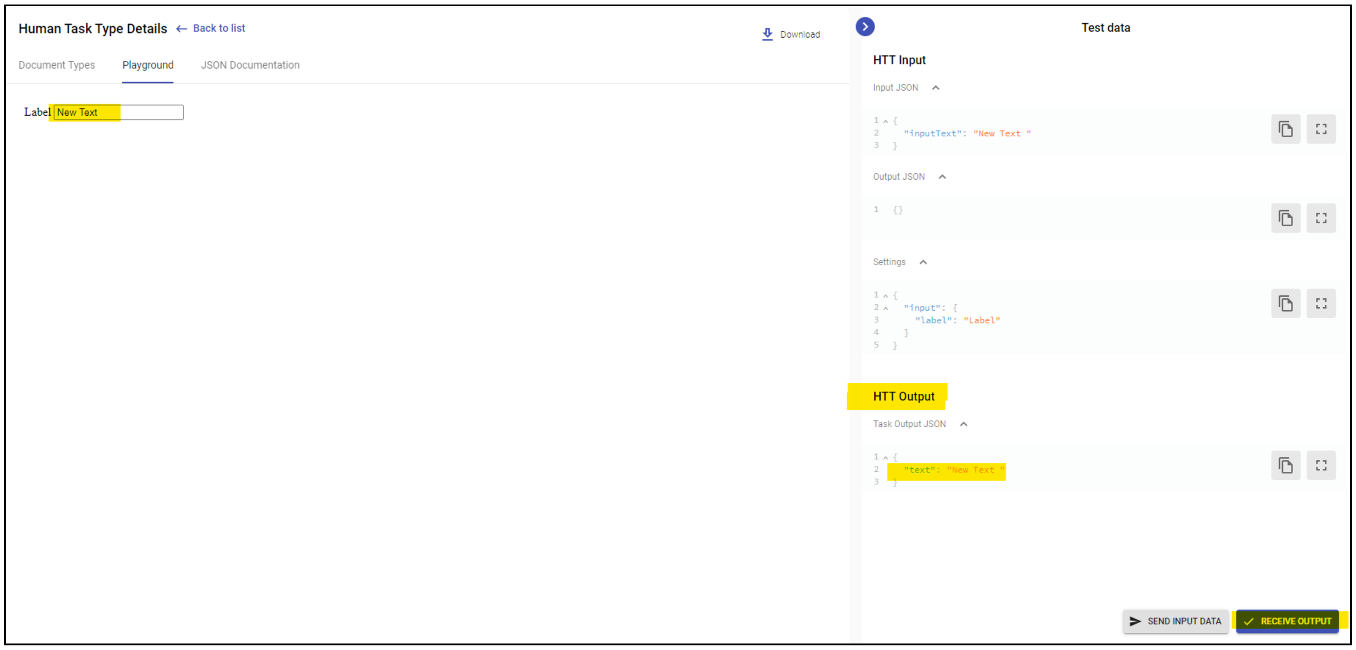
Settings:

```
{
   "input": {
     "label": "Label"
   }
}
```

Task Output JSON:

```
{}
```

To verify that HTT properly sends output on Workspace you need just click "RECEIVE OUTPUT" button. After that "HTT Output" section will be immediately changed.



Test data

HTT Input

Input JSON

```
{
  "inputText": "New Text "
}
```

Output JSON

```
{}
```

Settings

```
{
  "input": {
    "label": "Label"
  }
}
```

HTT Output

Task Output JSON

```
{
  "text": "New Text "
}
```

Full Example

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Human Task Example</title>
</head>
<body>
    <div id="root">
        <div class="app">
            <label for="main-input" id="main-label"></label>
            <input type="text" id="main-input">
        </div>
    </div>
</body>
<script>
    function submit() {
        const inputValue = document.getElementById('main-input').value;
        const outputJson = { text: inputValue };
        window.parent.postMessage(outputJson, '*');

        const data = {
            'validation-message': true,
            result: true,
        };
        window.parent.postMessage(data, '*');
    }

    function renderApp(data) {
        const template = document.createElement('div');
        const root = document.getElementById('root');
        root.appendChild(template);

        const input = document.getElementById('main-input');
        input.value = data.inputJson.inputText;

        const inputLabel = document.getElementById('main-label');
        inputLabel.innerHTML = data.settings.input.label;
    }

    window.addEventListener('message', message => {
        const { data } = message;
        if (!data['validation-message']) {
            renderApp(data);
        } else if (data['validation-message']) {
            submit(data);
        }
    });
</script>
</html>
```

sample.json

```
{
  "inputJson": {
    "inputText": "Text"
  },
  "outputJson": {

  },
  "settings": {
    "input": {
      "label": "Label"
    }
  }
}
```

package.json

```
{
  "name": "human-task-type-example",
  "version": "0.1.0"
}
```

(v. 2.2) Run Human Task

You're in charge of controlling thread pool size

As the Automation Process isn't finished after you create Human Task, it waits for results to continue processing. It means that Java Thread is used at that time.

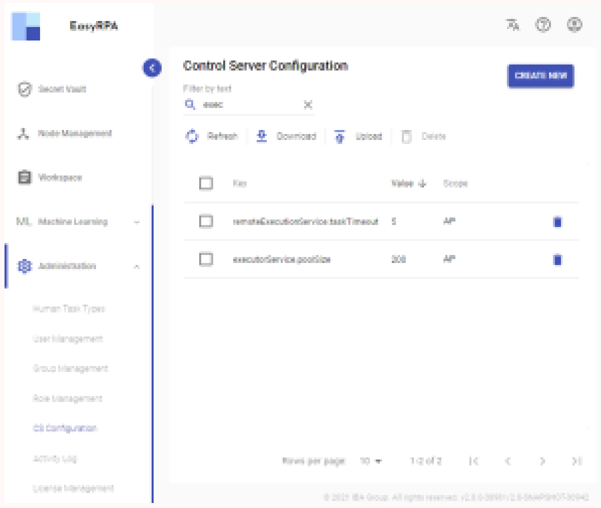
You should always develop your automation process keeping in mind that you can have some active (uncompleted) Human Tasks in parallel, so you should predict the possible amount of uncompleted Human Tasks.

If all threads from thread pool are in use - Automation Process won't be able to continue working with the different steps, because all threads are occupied by Human Tasks.

There're 2 Automation Process parameters you can control:

- **executorService.poolSize** - to specify thread pool size
- **remoteExecutionService.taskTimeout** - to specify the timeout for expiring of Human Tasks

You can find the default values on page **Administration CS Configuration:**



Control Server Configuration

Filter by text: exec

	Key	Value	Scope
	remoteExecutionService.taskTimeout	5	AP
	executorService.poolSize	200	AP

Rows per page: 10   1-2 of 2

Prepare input data

Before launching Human Task you must prepare an object of *HumanTaskData* and provide this bean as output from your own *Task* class.
As an example, let's look at the *PrepareInputApTask* java class below. It is a simple ApTask which creates and provides *HumanTaskDat a* as an output.

```
@ApTaskEntry(name = "Prepare Human Task Input")
@Slf4j
public class PrepareInputApTask extends ApTask {

        @Output(HumanTask.HUMAN_TASK_DATA_KEY)
        private HumanTaskData humanTaskData;

        @Override
        public void execute(){
                this.humanTaskData = new HumanTaskData();
                this.humanTaskData.setInputJson(new HashMap<>());
                this.humanTaskData.setDescription("Human Task Example");
                this.humanTaskData.setDocumentType("example");
                this.humanTaskData.setName("Human Task");
                this.humanTaskData.setPriority(1);
                log.info("Injecting task into workspace {} ", humanTaskData);
}
```

Execute Human Task

Now let's look how to set input and execute HumanTask from your ApModule class.

```
public TaskOutput run() throws Exception {
    return execute(getInput(), PrepareInputApTask.class)
                                .thenCompose(HumanTask.class)
                                .thenCompose(ProcessHumanTaskResultsTask.class);
}
```

Work with Human Task results

When Human Task will be completed you may need to handle output results. Bellow an example how to obtain and work with Human Task results.

```
@ApTaskEntry(name = "Working with Human Task results.")
@Slf4j
@InputToOutput
public class ProcessHumanTaskResultsTask extends ApTask {

        @Input(HumanTask.HUMAN_TASK_DATA_KEY)
        private HumanTaskData humanTaskData;

        @Override
        public void execute() throws Exception {
                log.info("Received response from Human Task {} ", humanTaskData.getTaskUuid());
                Map outputJson = humanTaskData.getOutputJson();
                //work with output..
        }
}
```

(v. 2.2) Digitizing Documents (OCR)

EasyRPA supports automation of process that requires electronic conversion of images of printed text into machine-encoded text.

The platform includes built-in OCR engine:

- Out-of-the-box solution: there is no necessity to write or maintain any custom code
- No OCR page limits per year
- Easy PDF automation
- Based on the leading open-source OCR engine
- Can recognize more than 100 languages "out of the box"

In addition, EasyRPA supports integration with ABBYY FlexiCapture software.

(v. 2.2) OCR Task

OCR Overview

OCR (Optical Character Recognition) is the electronic conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document or from subtitle text superimposed on an image (for example, from a television broadcast).
OCR of input documents is a separate step of the development process - usually, this is the conversion from PDF into Text documents. The **input quality** of original documents influences a lot of the output of this process.
EasyRPA uses Tesseract OCR, which is integrated into separate task and proposes an OOTB (out-of-the-box) solution: there is no necessity to write or maintain any custom code.

OCR Task Usage

Now, let us give you a few steps necessary to implement this approach.

1. Define **OcrTaskData** as output in your **Task** class:

   ```
   @Output(OcrTask.OCR_TASK_DATA_KEY)
   private OcrTaskData ocrTaskData;
   ```

   OcrTask.OCR_TASK_DATA_KEY - input/output data key. *OcrTask* uses this key to get input data for OCR.

2. Prepare OCR configuration providing S3 bucket name, Tesseract options and Image Magick options:

   ```
   Map configuration = new HashMap<String, Object>();
   configuration.put("bucket", "data");
   configuration.put("tesseractOptions", Arrays.asList("-l", "eng","--psm","3","--oem","3","--dpi", "800"));
   configuration.put("imageMagickOptions", Arrays.asList("-resample", "450", "-density", "350", "-quality",
   "100", "-background" , "white" , "-alpha" , "flatten"));
   configuration.put("hocrFixWords", Map.of("S(?=[0-9]+)", "\\$"));
   ```

   documents_bucket - the S3 bucket name which OCR will use to save results.
   tesseractOptions - Tesseract command line options. "-l" and "eng" means language = eng etc. Please see the external documentation Tesseract Command Line Usage.
   imageMagickOptions - OcrTask uses ImageMagick tool to split pdf by pages and print as images. You can provide a command line options to change some command line arguments. Please follow the ImageMagick Command Line documentation.
   hocrFixWords - HocrPostProcessor uses this config to fix OCR mistakes. Postprocessor accepts regular expression and replaces all matches by value. Pay attension that dollar sign and backslash(\ ) should be escaped in value string.

3. To initialize **OcrTaskData** you need to pass the S3 path to the particular document, add formats which will be used for getting OCR results and provide OCR configuration:

   ```
   this.ocrTaskData = new OcrTaskData();
   ocrTaskData.setDocumentLocation("ie_demo_invoice/invoice101.pdf");
   ocrTaskData.getFormats().addAll(Arrays.asList(OcrFormats.TEXT, OcrFormats.HOCR, OcrFormats.JSON,
   OcrFormats.IMAGE));
   ocrTaskData.setConfiguration(configuration);
   ```

   document location - the document path on S3 Storage.
   OcrFormats - the list of OCR formats which OCR Task is currently supported.
   configuration - the configuration map which you must set from the 2nd item.

4. Call **OcrTask:**

```
TaskOutput ocrInput = execute(getInput(), PrepareOcrTask.class).get();
TaskOutput ocrOutput = execute(ocrInput, OcrTask.class).get();
```

5. To get OCR use following example:

```
@ApTaskEntry(name = "Store OCR Result")
@Slf4j
@InputToOutput
public class StoreOcrResult extends ApTask {

        @Input(OcrTask.OCR_TASK_DATA_KEY)
        private OcrTaskData ocrTaskData;

        @Override
        public void execute() {
                log.info("Received response OCR Task {} ", ocrTaskData.getTaskUuid());
                List<OcrResult> ocrResults = this.ocrTaskData.getOcrResults();
                //...
        }

}
```

OcrTask.OCR_TASK_DATA_KEY - input/output data key. *OcrTask* uses this key to get input data for OCR.

Full example

PrepareOcrTask.java

```
import eu.ibagroup.easyrpa.engine.annotation.AfterInit;
import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.annotation.InputToOutput;
import eu.ibagroup.easyrpa.engine.annotation.Output;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrFormats;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrTask;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrTaskData;
import eu.ibagroup.easyrpa.utils.MlTaskUtils;
import eu.ibagroup.sample.ml.ie.repository.DocumentRepository;
import lombok.extern.slf4j.Slf4j;

import javax.inject.Inject;
import java.util.Arrays;
import java.util.Map;

@ApTaskEntry(name = "Prepare OCR Task")
@Slf4j
@InputToOutput
public class PrepareOcrTask extends ApTask {

    @Inject
    private DocumentRepository documentRepository;

    @Output(OcrTask.OCR_TASK_DATA_KEY)
    private OcrTaskData ocrTaskData;

    @Output("configuration")
    private Map<String, Object> configuration;

    @AfterInit
    public void init() {
        configuration = MlTaskUtils.readObjectFromString(getConfigurationService().get("CONFIGURATION", "{}"),
Map.class);
    }

    @Override
    public void execute() {
        this.ocrTaskData = new OcrTaskData();
        ocrTaskData.setDocumentLocation("ie_demo_invoice/invoice101.pdf");
        ocrTaskData.getFormats().addAll(Arrays.asList(OcrFormats.TEXT, OcrFormats.HOCR, OcrFormats.JSON,
OcrFormats.IMAGE));
        ocrTaskData.setConfiguration(configuration);
    }
}
```

StoreOcrResult.java

```
package eu.ibagroup.sample.ml.ie.tasks;

import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.annotation.Input;
import eu.ibagroup.easyrpa.engine.annotation.InputToOutput;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrResult;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrTask;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrTaskData;
import lombok.extern.slf4j.Slf4j;

import java.util.List;

@ApTaskEntry(name = "Store OCR Result")
@Slf4j
@InputToOutput
public class StoreOcrResult extends ApTask {

    @Input(OcrTask.OCR_TASK_DATA_KEY)
    private OcrTaskData ocrTaskData;

    @Override
    public void execute() {
        log.info("Received response OCR Task {} ", ocrTaskData.getTaskUuid());
        List<OcrResult> ocrResults = this.ocrTaskData.getOcrResults();
        //...
    }

}
```

OcrProcess.java

```
import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModule;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import eu.ibagroup.easyrpa.engine.boot.ApModuleRunner;
import eu.ibagroup.easyrpa.engine.boot.configuration.DevelopmentConfigurationModule;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrTask;
import eu.ibagroup.sample.ml.ie.tasks.StoreOcrResult;
import eu.ibagroup.sample.ml.ie.tasks.PrepareOcrTask;
import lombok.extern.slf4j.Slf4j;

@ApModuleEntry(name = "OCR AP Sample", description = "OCR AP Sample")
@Slf4j
public class OcrProcess extends ApModule {

    public TaskOutput run() throws Exception {
                return execute(getInput(), PrepareOcrTask.class)
                        .thenCompose(execute(OcrTask.class))
                        .thenCompose(execute(StoreOcrResult.class)).get();
    }
    public static void main(String[] args) {
        ApModuleRunner.localLaunch(OcrProcess.class, new DevelopmentConfigurationModule("cs.json"));
    }

}
```

(v. 2.2) ABBYY FlexiCapture integration module

What is FlexiCapture?

ABBYY FlexiCapture is an Intelligent Document Processing platform built for complex digital document processing. FlexiCapture brings together the best NLP, machine learning, and advanced OCR capabilities into a single, enterprise-scale platform to handle every type of document, from simple forms to complex free-form documents. This library available in EasyRPA nexus by:

```
<dependency>
    <groupId>eu.ibagroup</groupId>
    <artifactId>easy-rpa-flexicapture-client</artifactId>
    <version>1.1</version>
</dependency>
```

Preconditions

- Installed and configured FlexiCapture server v.12 or newer
- To have valid license, including Verification and Scanning stations features
- FlexiLayout project should be created and deployed to the server. FlexiCapture users and developers materials can be found on the official ABBYY FlexiCapture tutorial resource https://help.abbyy.com/en-us/flexicapture/12/flexilayout_studio/tutorial_all

Module usage

Git repository location

https://git.icdc.io/easyrpa/easy-rpa-libs/easy-rpa-flexicapture-client

Maven

To include the module, use maven dependency:

```
        <dependency>
                <groupId>eu.ibagroup</groupId>
                <artifactId>easy-rpa-flexicapture-client</artifactId>
                <version>1.0</version>
        </dependency>
```

Development steps

Below provided a list and sequence of methods that are required to implement attended data extraction scenario:

Service Initialization

```
FlexiCaptureService client = new FlexiCaptureService(String flexiCaptureBaseUrl, String username, String
password);
```

Recognition task creation

```
SessionContainer createRecognitionTask(String fileName, InputStream stream, String projectName);
```

Get task completion percentage

```
int percentCompleted = getTaskStatus(@NotNull SessionContainer recognitionTaskData)
```

Get list of documents that need verification

```
List<String> getVerificationQueue(SessionContainer recognitionTaskData)
```

Build task verification link

```
{FlexiCaptureServerUrl}/FlexiCapture12/Verification/Verify?taskId={taskId}
```

Get recognition results

```
ExtractResultsContainer getBatchExtractResult(SessionContainer recognitionTaskData)
```

Usage Examples

Synchronous call

```
    public void runAttendedScenario() throws  Exception{
        String fileNameWithExtension = "ES_Manzana.tif";
        String invoiceDemoProject = "MultipageInvoiceProject";
                String username = "Administrator";
                String password = "password";

        FlexiCaptureService client = new FlexiCaptureService("http://10.224.32.19", username, password);
        InputStream fileStream = this.getClass().getResourceAsStream(fileNameWithExtension);

        String fileName = FilenameUtils.getName(fileNameWithExtension);

        SessionContainer task = client.createRecognitionTask(fileName, fileStream, invoiceDemoProject);
        int percentCompleted = 0;
        while (percentCompleted < 100){
            percentCompleted = client.getTaskStatus(task);
            logger.info("processed " + percentCompleted + "%");
            if(percentCompleted < 100){
                if(client.getVerificationQueue(task).size() > 0){
                    logger.warn("Manual verification required: " + client.getVerificationUrl() + task.
getTaskId());
                }
                Thread.sleep(5000);
            }
        }
        ExtractResultsContainer extractResults = client.getBatchExtractResult(task);
        List<File> savedFiles = CommonUtils.storeToDir(extractResults);
        logger.info("Recognition results:");
        for(File savedFile : savedFiles){
            logger.info(savedFile);
        }
    }
```

Asynchronous call

```
public class FlexiCaptureEventListenerImpl implements FlexiCaptureEventListner {
    private static final Logger logger = Logger.getLogger(FlexiCaptureEventListenerImpl.class);
    @Override
    public void onSuccess(ExtractResultsContainer extractResults, SessionContainer task) throws Exception {
        List<File> savedFiles = CommonUtils.storeToDir(extractResults);
        savedFiles.forEach(file -> logger.info(file));
    }

    @Override
    public void onError(Exception e) {
        logger.error("Document recognition error: ", e);
    }

    @Override
    public void onValidationRequired(String verificationUrl, SessionContainer task) {
        System.out.println("Manual verification required: " + verificationUrl + task.getTaskId());
    }
}

public class FlexiCaptureAsyncRunner {
    public static void main(String[] args) throws UserNotFoundException, ServerFault, MalformedURLException {
        String fileNameWithExtension = "/ES_Manzana.tif";
        String invoiceDemoProject = "InvoiceDemoProject";
        String username = "Administrator";
        String password = "Start123";

        InputStream fileStream = FlexiCaptureAsyncRunner.class.getResourceAsStream(fileNameWithExtension);
        FlexiCaptureEventListner listener = new FlexiCaptureEventListenerImpl();
        FlexiCaptureService client = new FlexiCaptureService("http://10.224.32.19", username, password);
        String fileName = FilenameUtils.getName(fileNameWithExtension);

        client.createAsyncRecognitionTask(fileName, fileStream, invoiceDemoProject, listener);

    }
}
```

(v. 2.2) OCR Best Practices

OCR (Optical Character Recognition) is the technology used to recognize text inside images, such as scanned documents or photos, and convert it into machine-readable text data.

The built-in OCR Flow of the EasyRPA Platform includes 3 main steps:

- ImageMagick Pre-processor
- Tesseract OCR Engine
- HOCR Post-processor

ImageMagick Pre-processor

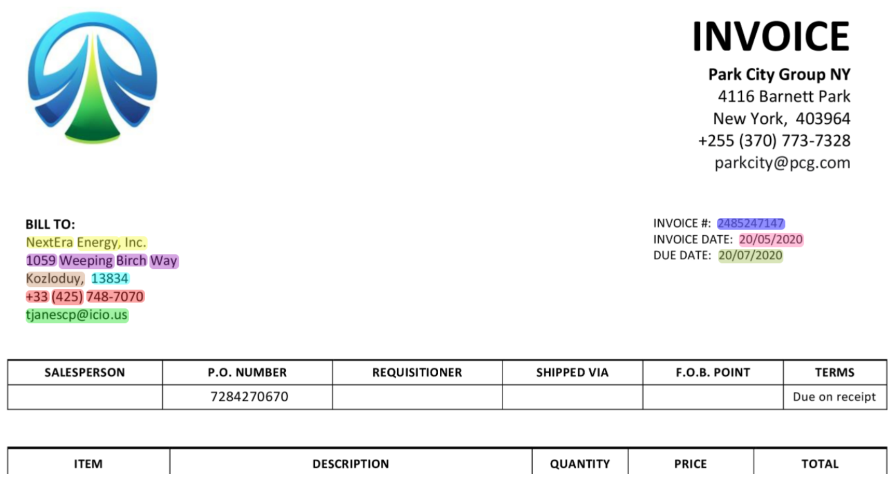
ImageMagick Pre-processor is used for displaying, creating, converting, modifying, and editing digital images. ImageMagick mainly consists of a number of options for manipulating images.

The list of ImageMagick options recommended for use:

- **-resample**. This option resizes the image so that its rendered size remains the same as the original at the specified target resolution. Recommended value: at least 300 DPI. For more details about DPI refer to Tesseract documentation - Rescaling.
- **-density**. This option specifies the image resolution to store while encoding a raster image. The default unit of measure is in dots per inch (DPI). The recommended value should be equal to the *-resample* option.
- **-units**. This option specifies the units of image resolution and is normally used in conjunction with the *-density* option. Recommended value: "*PixelsPerInch*."
- **-quality**. This option specifies the JPEG/MIFF/PNG compression level. Recommended value: "*100*".
- **-deskew**. This option straightens an image. A threshold of 40% works for most images. Recommended value: "*40%".*
- **-contrast**. Use this option to enhance the image contrast.
- **+contrast**. Use this option to reduce the image contrast.
- **-alpha.** This option helps to fix the layers in the readable PDFs. Recommended value: "*flatten*".

Below you can see the output result of the readable PDF recognition with and without "-alpha", "flatten" correspondingly:

1) with:



INVOICE

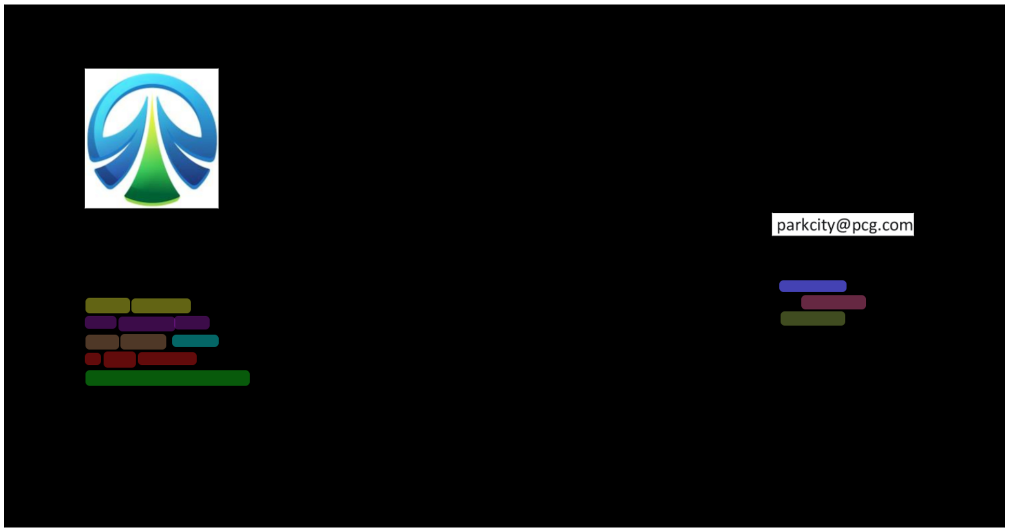
Park City Group NY
4116 Barnett Park
New York, 403964
+255 (370) 773-7328
parkcity@pcg.com

BILL TO:
NextEra Energy, Inc.
1059 Weeping Birch Way
Kozloduy, 13834
+33 (425) 748-7070
tjanescp@icio.us

INVOICE #: 2485247147
INVOICE DATE: 20/05/2020
DUE DATE: 20/07/2020

SALESPERSON	P.O. NUMBER	REQUISITIONER	SHIPPED VIA	F.O.B. POINT	TERMS
	7284270670				Due on receipt

ITEM	DESCRIPTION	QUANTITY	PRICE	TOTAL

2) without: 

For an extended list of ImageMagick options, refer to the Full List of ImageMagick Options.

Tesseract OCR Engine

Tesseract is an OCR engine with support for unicode and the ability to recognize more than 100 languages out of the box.

The list of Tesseract options recommended for use:

- **-l**. This option specifies the language or script to use. If none is specified, 'eng' (English) is assumed. More than one language or script may be specified by using **+**. Example: "eng+deu+fra". For more information, you can refer to the Full List of Tesseract Languages.
- **--psm**. Page Segmentation Method sets Tesseract to only run a subset of layout analysis and assume a certain form of an image. Recommended value: *12* (Sparse text with OSD).
- **--oem**. This option specifies OCR Engine Mode. Recommended value: *3* (Default, based on what is available).
- **--dpi**. This option specifies the resolution *N* in DPI for the input image. The recommended value should be equal to the -*resample* ImageMagick option and at least 300 DPI.

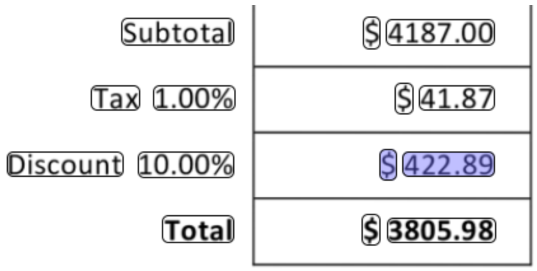
⚠️ Without this option, the resolution is read from the metadata included in the image. If an image does not include that information, Tesseract tries to guess it, which leads to a significant increase in document processing time.

For more information about Tesseract options, visit Tesseract Manual Page.

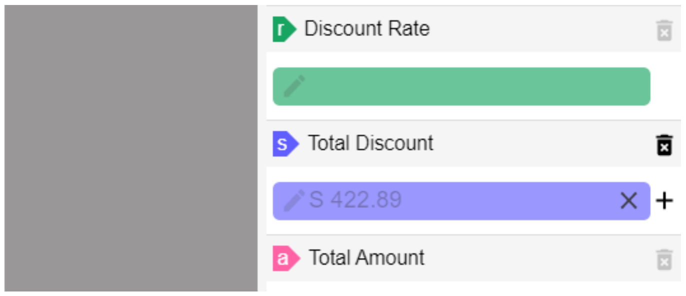
HOCR Post-processor

HOCR post-processor helps to fix common OCR mistakes by providing pairs of recognized and corrected words. Post-processing rules can be set either as pairs of words or as a regexp function.

In the provided example it is seen that '$' was recognized as 'S'.



Subtotal		$ 4187.00
Tax	1.00%	$ 41.87
Discount	10.00%	$ 422.89
Total		$ 3805.98



It is possible to fix the OCR mistake using:

- pair of values: "^S$": "\\$" - in case '$' represents a separate bbox.

- or regexp function : "S(?=[0-9]+([.,][0-9]+)?)": "\\$"- in case '$' represents a part of bbox and is followed by the amount, i.e. '$422.89'.

(v. 2.2) Machine Learning (ML)

(v. 2.2) Concepts and Entities

Machine learning (**ML**) is a method of data analysis that automates analytical model building. Machine learning algorithms build a model based on sample data, known as **training data**, in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

EasyRPA platform provides developers with infrastructure to train ML models and use for data processing.

Let's look at EasyRPA artifacts related to Machine Learning area.

Document Sets

Work on ML models starts with data - preferably, lots of data (documents) for which the target answer known. When the target answer is assigned the documents are called **labeled** or **tagged data**. For example, for the email classification problem, the target is a label that indicates whether an email is spam or not spam. The ML algorithm teaches itself to learn from the labeled examples that we provide.

Often, data is not readily available in a labeled form. Collecting and labeling data is often the most important step in solving an ML problem. The example data should be representative of the data that you will have when you are using the model to make a prediction. For example, if you want to predict whether an email is spam or not, you must collect both positive (spam emails) and negative (non-spam emails) for the machine learning algorithm to be able to find patterns that will distinguish between the two types of email.

Once you have the labelled data, you might need to convert it to a format that is acceptable to your algorithm or software - a file called **training set**. The next step is actually **train a model**. This process invokes the ML algorithm with training data and all the parameters and configurations required, as the result ML model is created and training log is stored in the system.

Document Processor

Document Processor is a special automation process that controls life-cycle of document sets and handles all data transformations required to collect data and apply OCR, label documents in Workspace, prepare training data sets and train models.

ML Models

The term **ML Model** refers to the model artifact that is created by the training process. In EasyRPA, ML Model is an archive file that includes all the scripts, serialized classes, parameters, and configuration files required to run the data processing pipeline.

EasyRPA platform supports two types of ML models:

- (v. 2.2) Classification Models
- (v. 2.2) Information Extraction Models

Model Repository

Model Repository stores all the ML models available in the system: whenever a new model is trained it is placed into the repository, whenever a model is required to process data it is retrieved from the repository. Each model is uniquely identified by its **name** and **versi on**.

EasyRPA Control Server also provides a way to export existing models as a file and import pre-trained models into the system.

ML Container

ML Container is a component of EasyRPA platform that is responsible for two main tasks:

- When performing **train** request model container is provided with a training set, model configuration files that specify model type and training parameters. ML Container selects training algorithm suitable for the specified model type, performs model training process for the specified number of iterations, selects the best model and packages it along with the required configuration files. On the last step ML container upload the resulting model package into the model repository.
- Whenever a model is used to **process data**, the request specifies a model to be executed and input JSON. ML Container retrieves the specified model from the model repository, if required, and executes it. The result of execution is placed in output JSON.

EasyRPA Control Server communicates with ML Container via a message queue.

ML Container is one of the platform **scalable components**: there is an option to scale number of ML Containers in deployment, if required.

(v. 2.2) Classification Models

Overview

Document classification or **document categorization** is a problem in library science, information science and computer science. The task is to assign a document to one or more classes or categories. This may be done "manually" or algorithmically.

EasyRPA includes implementation of both **Single-label Classification** and **Multi-label Classification**.

Let's see the difference of these two methods on very simple example and assume that we have 5 categories of documents. No matter what type of classification is used, the model's response for a single document is always a vector of 5 values:

- Single-label classification - each category is assigned a value from 0 to 1; sum of all probabilities equals 1
- Multi-label classification - each category is assigned a value from 0 to 1 independently of the others; therefore, sum of all probabilities is in 0..5 range

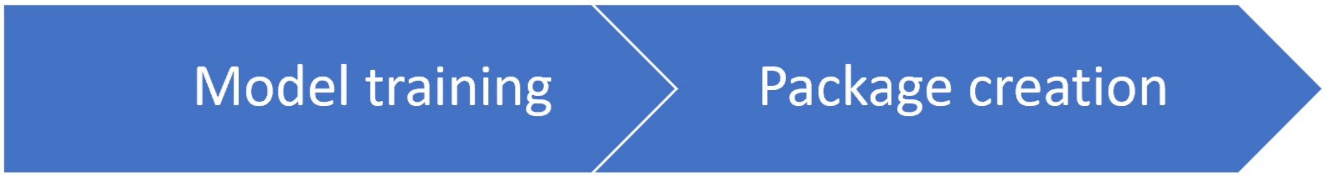
For example, we classify articles by three categories: *sports*, *business*, *press*. An article about a sports magazine may potentially have the following score:

- Single-label classification - [sports = 0.3, business = 0.0, press = 0.7]
- Multi-label classification - [sports = 0.8, business = 0.0, press = 0.9]

Document Classification as a Pipeline

Let's look in more details, which stages are included into both processes: model training and execution.

Model Training Process



Model training

On this step EasyRPA trains ML model using the training set provided. The system automatically shuffle and splits the provided set into training and validation subsets, runs training for specified number of iterations, and select the best model.

Process developer can specify model type (single vs multi label) and number of iterations using a configuration JSON file.

Package creation

The best Spacy model comes packaged along with configuration files and uploaded to the models repository.

Classification Process

Model execution

The classification model is executed for a single document.

The result of model execution is a JSON with two sections

- **scores** section contains key-value pairs with category names and their probabilities
- **categories** section contains a single category, in case of single-label classification, and an empty list if it is a multi-label process.

```
{
  "scores": {
    "Science": 0.7,
    "Sport": 0.3
  },
  "categories": [
    "Science"
  ]
}
```

```
{
  "scores": {
    "Science": 0.1,
    "Sport": 0.8,
    "Business": 0.2,
    "Media": 0.6
  },
  "categories": []
}
```

Model Training Configuration File

To train a document classification model you need to provide a ZIP archive with files that defines configuration parameters for the training process.

Let's look at these configuration files in details.

model.json

Developer can specify model type (single vs multi label), number of training iterations and categories to train in *model.json* file. This file is **required**.

Default configuration values are:

- single_label=True
- iterations=10
- categories=[] (Categories are extracted from train data if not specified)

```
{
 "model_type": "textcat",
 "train_config": {
  "single_label": true,
  "iterations": 5,
  "categories": [
    "Debit Note",
    "Invoice"
  ]
 }
}
```

(v. 2.2) Information Extraction Models

Overview

Information extraction (IE) is the automated retrieval of specific **information** related to a selected topic from a body or bodies of text. **In formation extraction** tools make it possible to pull **information** from text documents, databases, websites or multiple sources.

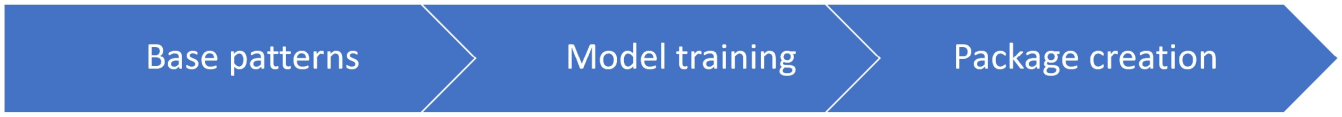
EasyRPA provides infrastructure to create and run machine learning models that extract information from PDF documents.

Information Extraction as a Pipeline

Information Extraction process is implemented in EasyRPA as a pipeline. ML models is main, but not single step in this pipeline: platform also provides several extension points to extend ML with rules and dictionaries.

Let's look in more details, which stages are included into both process: model training and execution.

Model Training Process



Model training

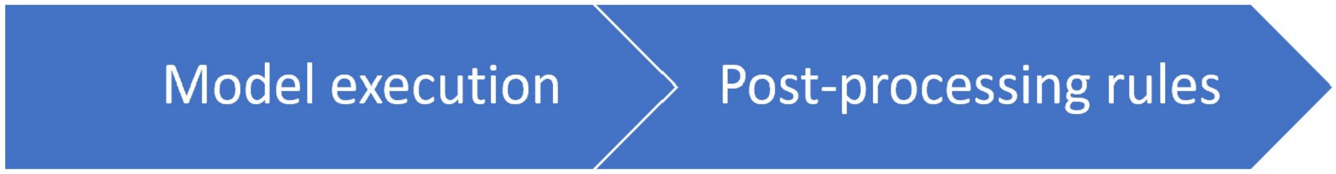
On this step EasyRPA trains ML model using the training set provided. The system automatically shuffle and splits the provided set into training and validation subsets, runs training for specified number of iterations, and select the best model.

Process developer can specify model type using a configuration JSON file.

Package creation

The best Spacy model comes packaged with meta.json, model.json and configuration files and uploaded to Nexus repo.

Information Extraction Process



Model execution

The model is executed for a single document.

The result of model execution is entities list. Each entity consists of label name, count index, label content and a list of OCR words that match the entity region.

```
{
  "entities": [
    {
      "name": "DebitNoteId",
      "words": [
        {
          "bbox": [
            0.8227450980392157,
            0.1696969696969697,
            0.9168627450980392,
            0.18262626262626264
          ],
          "id": "page0_area2_paragraph2_line3_word15",
          "page": 0,
          "content": "DNT6268231"
        }
      ],
      "index": 0,
          "score": 1.0,
      "content": "DNT6268231"
    },
    {
      "name": "DebitNoteDate",
      "words": [
        {
          "bbox": [
            0.8227450980392157,
            0.19818181818181818,
            0.92,
            0.21151515151515152
          ],
          "id": "page0_area2_paragraph2_line4_word23",
          "page": 0,
          "content": "2018-06-30"
        }
      ],
      "index": 0,
          "score": 0.97,
      "content": "2018-06-30"
    }
  ]
}
```

Model Training Configuration File

To train a Information Extraction model you need to provide a ZIP archive with files that defines configuration parameters for the training process.

Let's look at these configuration files in details.

model.json

One of the **required** config files is a model description with defined model type.

Train config is an optional section, so default *language* is "en" and number of *iterations* is 30.
Process config is also optional, default *concat_single_entities* value is true.

```
{
 "model_type": "ie",
 "train_config": {
  "lang": "de",
  "iterations": 5
 },
 "process_config": {
   "concat_single_entities": false
  }
}
```

labels.json

Labels from this configuration are added to the NER pipe at training stage. In case of empty configuration all labels found in the train dataset will be automatically added to the model, and the output dimension will be inferred automatically (expensive operation). The multiplicity flag affects how the entity index is calculated at processing stage. Index of labels with multiplicity equals *True* increments through the whole document while for labels with *False* multiplicity index is always zero. This file is **optional**.

```
{
  "DebitNoteDate": false,
  "DebitNoteId": false,
  "VendorSupportProgram": false,
  "Percent": false,
  "TotalAmount": false,
  "SKU": true,
  "SKUDescription": true,
  "Units": true,
  "Quota": true,
  "kwDebitNoteID": false,
  "kwDebitNoteDate" : false,
  "BillingFrequency" : false
}
```

base_model_patterns.json

Base model patterns are used to configure **EntityRuler** for labeling datum elements. It runs before fetching data and provides model with additional information on the document structure increasing accuracy of data extraction. This file is **optional**.

```
{
    "label": "kwDebitNoteID",
    "id": "kwDebitNoteID",
    "pattern": [
      { "TEXT": "Debit" },
      { "TEXT": { "REGEX": "^Note.*$" } },
      { "TEXT": "#" },
      { "TEXT": ":" }
    ]
}
```

post_processing_rules.json

After NER extraction model uses **EntityMatcher** with rules defined in post_processing_rules.json. Configuration file should contain a list of label names with regular expressions for searching for entities. This file is **optional**.

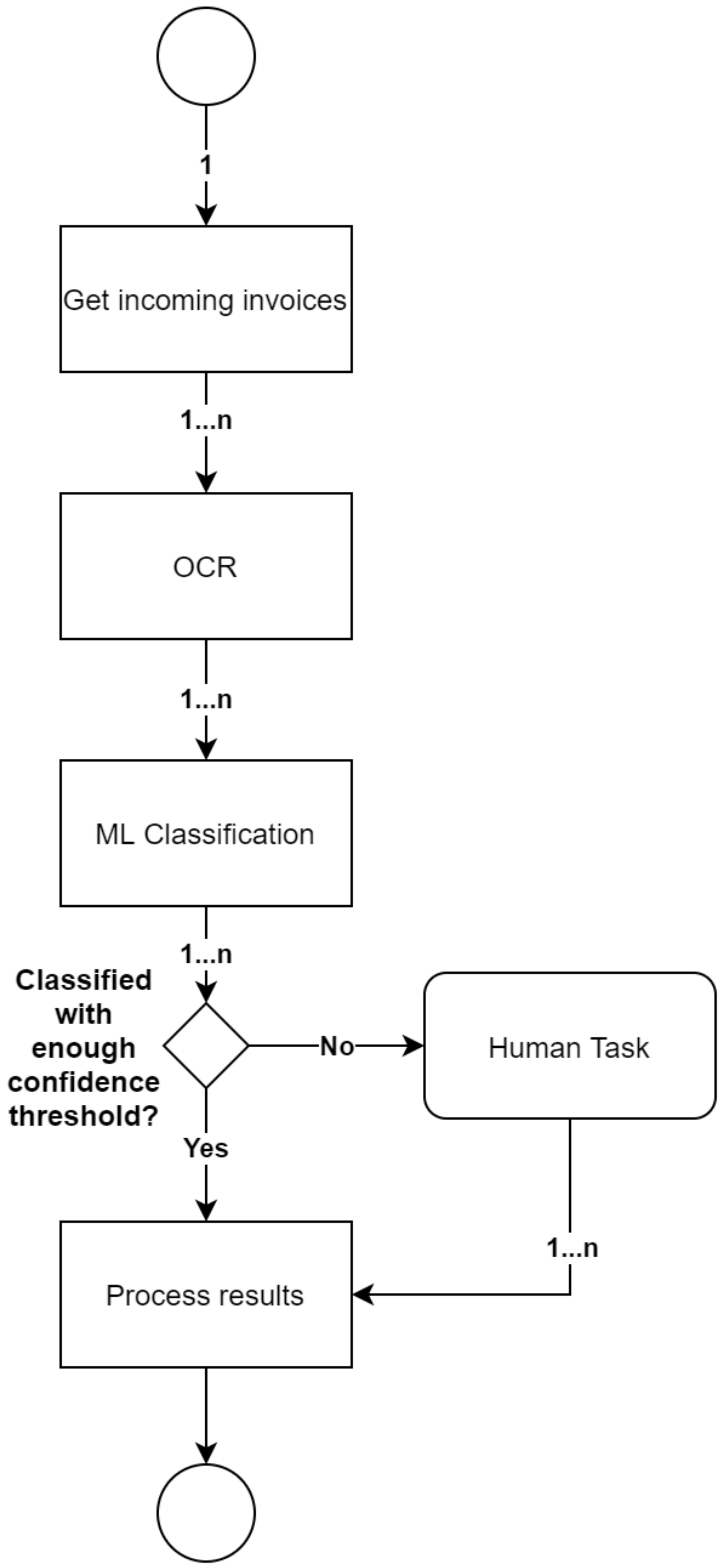
```
[
  {
    "label": "DebitNoteId",
    "regex": [
      "Debit Note#: (...\\d{7})"
    ]
  }
]
```

(v. 2.2) Document Classification process

This article covers the end-to-end process of developing Classification automation and contains 2 sections:

1. Setup Human Task and Train model **in 3 steps**
2. Develop Automation Process which use ML Classification model **in 5 steps**

From the high-level perspective, the Classification automation process looks like on the diagram below and it should process all records asynchronously in parallel:


1
Get incoming invoices
1...n
OCR
1...n
ML Classification
1...n
Classified with enough confidence threshold?
No
Human Task
Yes
1...n
Process results

(v. 2.2) Setup Human Task and Train ML model (Classification)

(v. 2.2) Step 1. Create new Document Type (Classification)

To describe which fields you want to extract from document you should create new **Document Type**.

Document Type is a configuration which lets Human Task and ML model know the output fields you want to receive from the document.

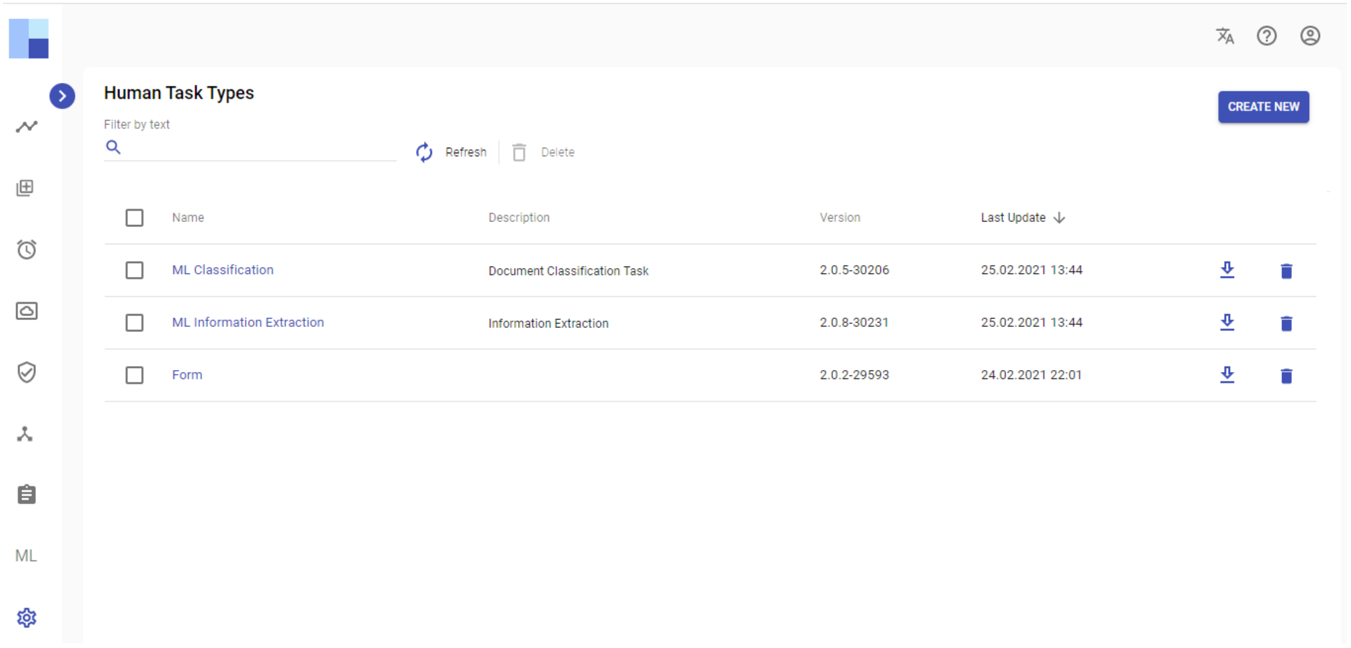
Each Document Type relates to some of the **Human Task Type**.

Human Task Type is a special small Web-application which can read and parse your Document Type configuration and it knows how to display input documents.

EasyRPA provides 3 predefined Human Task Types:

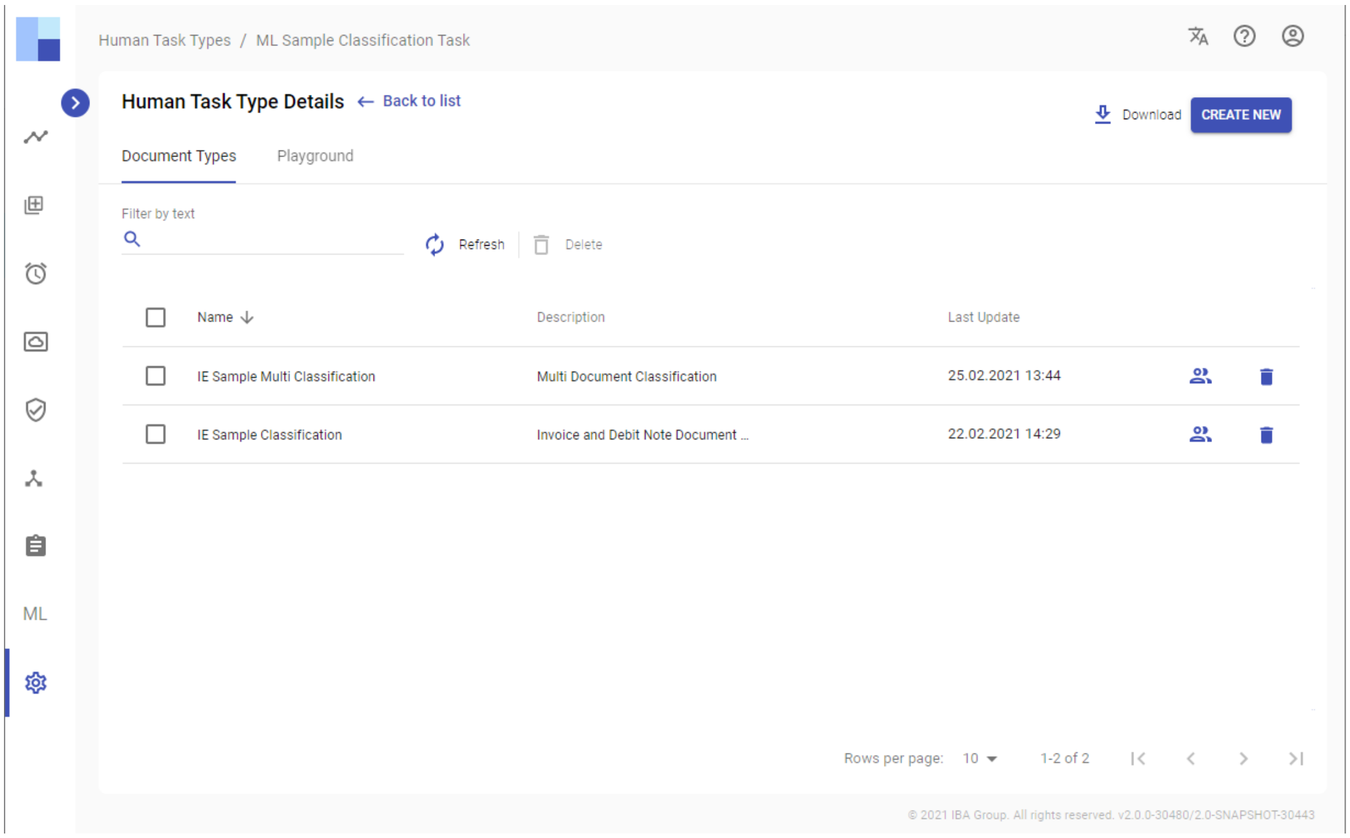
- Information Extraction
- Document Classification
- Form

For the Classification process we need to create Document Type which relates to the **ML Classification** predefined Human Task Type:



Name	Description	Version	Last Update
ML Classification	Document Classification Task	2.0.5-30206	25.02.2021 13:44
ML Information Extraction	Information Extraction	2.0.8-30231	25.02.2021 13:44
Form		2.0.2-29593	24.02.2021 22:01

When you open ML Classification type there's a list of all document types are related to that Human Task Type, and you will also see "CREATE NEW" button:


Human Task Types / ML Sample Classification Task

Human Task Type Details ← Back to list

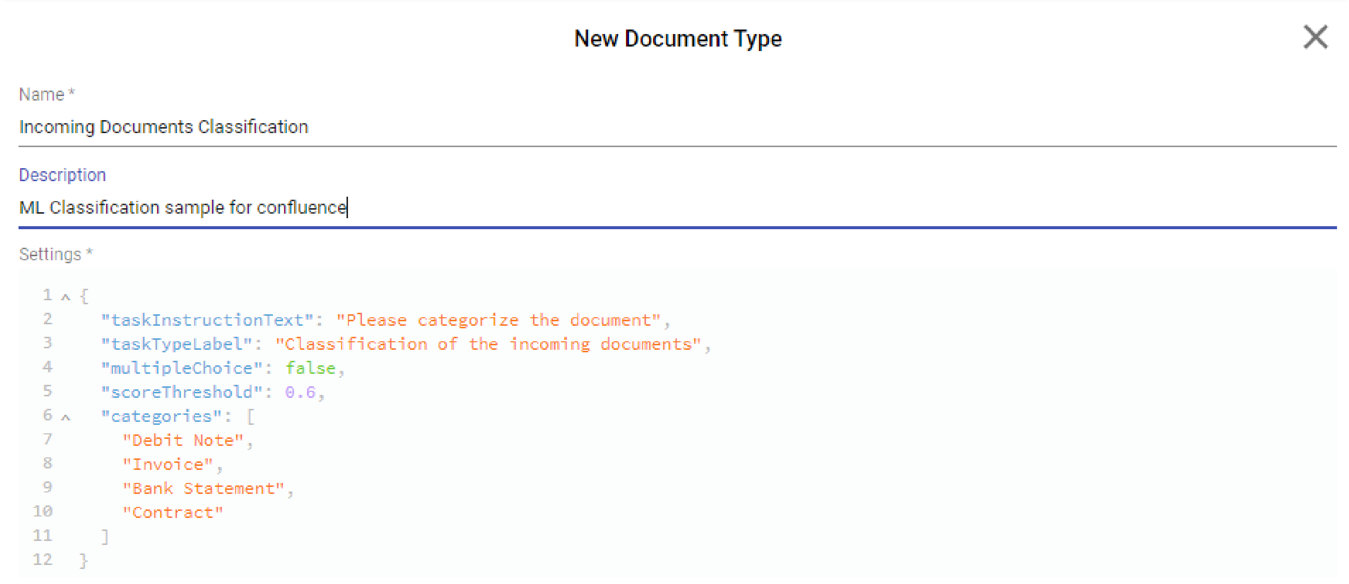
Document Types | Playground

	Name	Description	Last Update
☐	IE Sample Multi Classification	Multi Document Classification	25.02.2021 13:44
☐	IE Sample Classification	Invoice and Debit Note Document ...	22.02.2021 14:29

Rows per page: 10 1-2 of 2

© 2021 IBA Group. All rights reserved. v2.0.0-30480/2.0-SNAPSHOT-30443

You need to provide the name, description and settings for new Document Type.


New Document Type

Name *
Incoming Documents Classification

Description
ML Classification sample for confluence

Settings *
```
{
  "taskInstructionText": "Please categorize the document",
  "taskTypeLabel": "Classification of the incoming documents",
  "multipleChoice": false,
  "scoreThreshold": 0.6,
  "categories": [
    "Debit Note",
    "Invoice",
    "Bank Statement",
    "Contract"
  ]
}
```

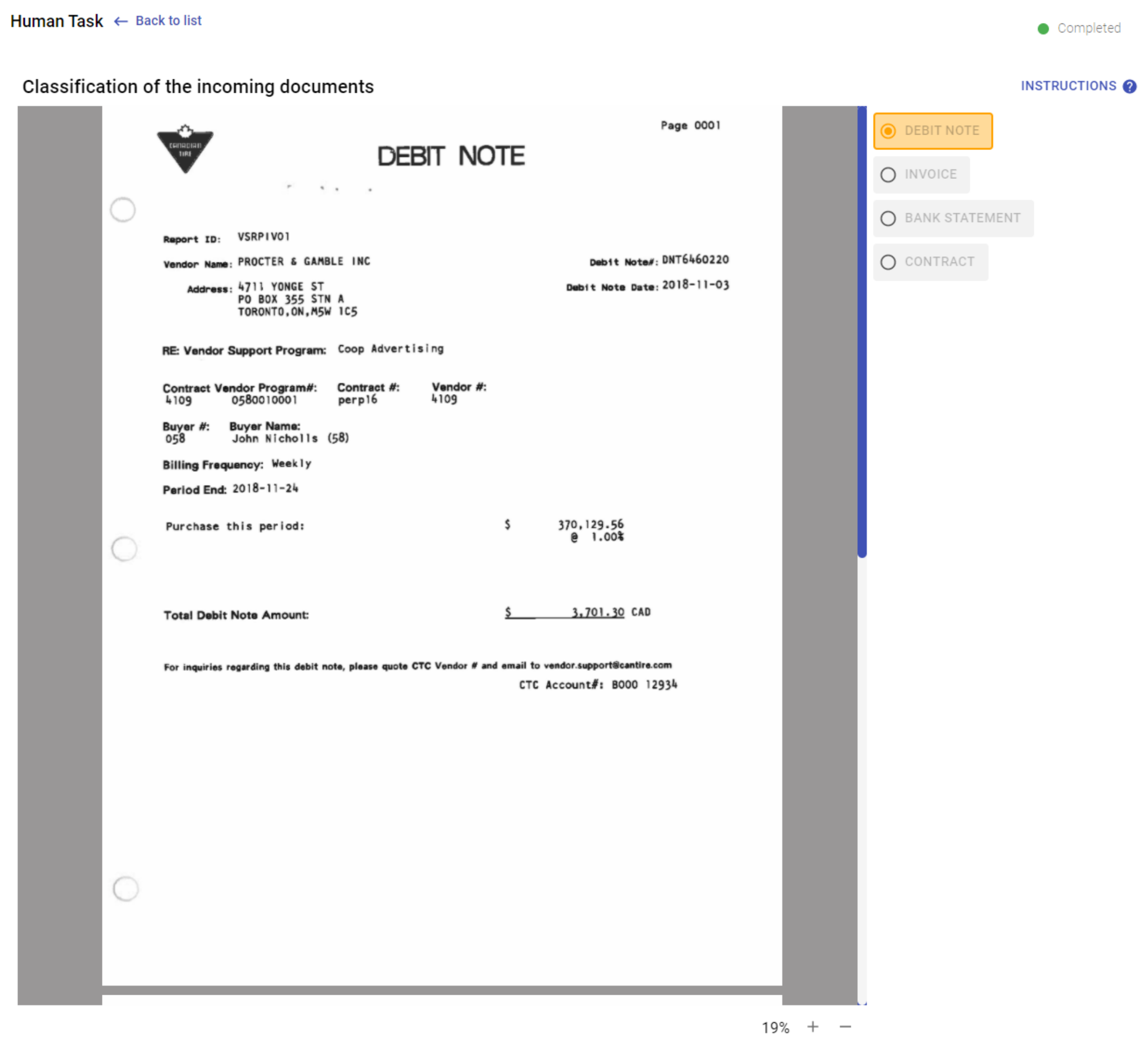
Settings is a configuration in JSON format which has the structure as in example bellow:

```
{
  "taskInstructionText": "Please categorize the document.",
  "taskTypeLabel": "Classification of the Document Type",
  "multipleChoice": false,
  "scoreThreshold": 0.6,
  "categories": [
    "Debit Note",
    "Invoice",
    "Bank Statement",
    "Contract"
  ]
}
```

Settings contains:

- **taskInstructionText** (string) - text of the instruction for workers who will process the document set
- **taskTypeLable** (string) - label which is shown as the main header of your task
- **multipleChoice** (boolean) - configure if documents may have 1 or multiple classes
- **scoreThreshold** (decimal) - is used when **multipleChoice is true** to auto select the categories after ML. When **multipleChoice is false**, the **scoreThreshold** doesn't matter because the category with highest score is automatically selected.
- **categories** (list of strings) - list of predefined classes your documents may relate to

The Document Type which is configured above will looks like on the following image:


Human Task ← Back to list

● Completed

Classification of the incoming documents

INSTRUCTIONS ?

Page 0001

DEBIT NOTE

Report ID: VSRPIV01

Vendor Name: PROCTER & GAMBLE INC

Address: 4711 YONGE ST
PO BOX 355 STN A
TORONTO,ON,M5W 1C5

Debit Note#: DNT6460220

Debit Note Date: 2018-11-03

RE: Vendor Support Program: Coop Advertising

Contract Vendor Program#:		Contract #:	Vendor #:
4109	0580010001	perp16	4109

Buyer #:	Buyer Name:
058	John Nicholls (58)

Billing Frequency: Weekly

Period End: 2018-11-24

Purchase this period: $ 370,129.56
@ 1.00%

Total Debit Note Amount: $ 3,701.30 CAD

For inquiries regarding this debit note, please quote CTC Vendor # and email to vendor.support@cantire.com

CTC Account#: B000 12934

[✓ DEBIT NOTE]

INVOICE

BANK STATEMENT

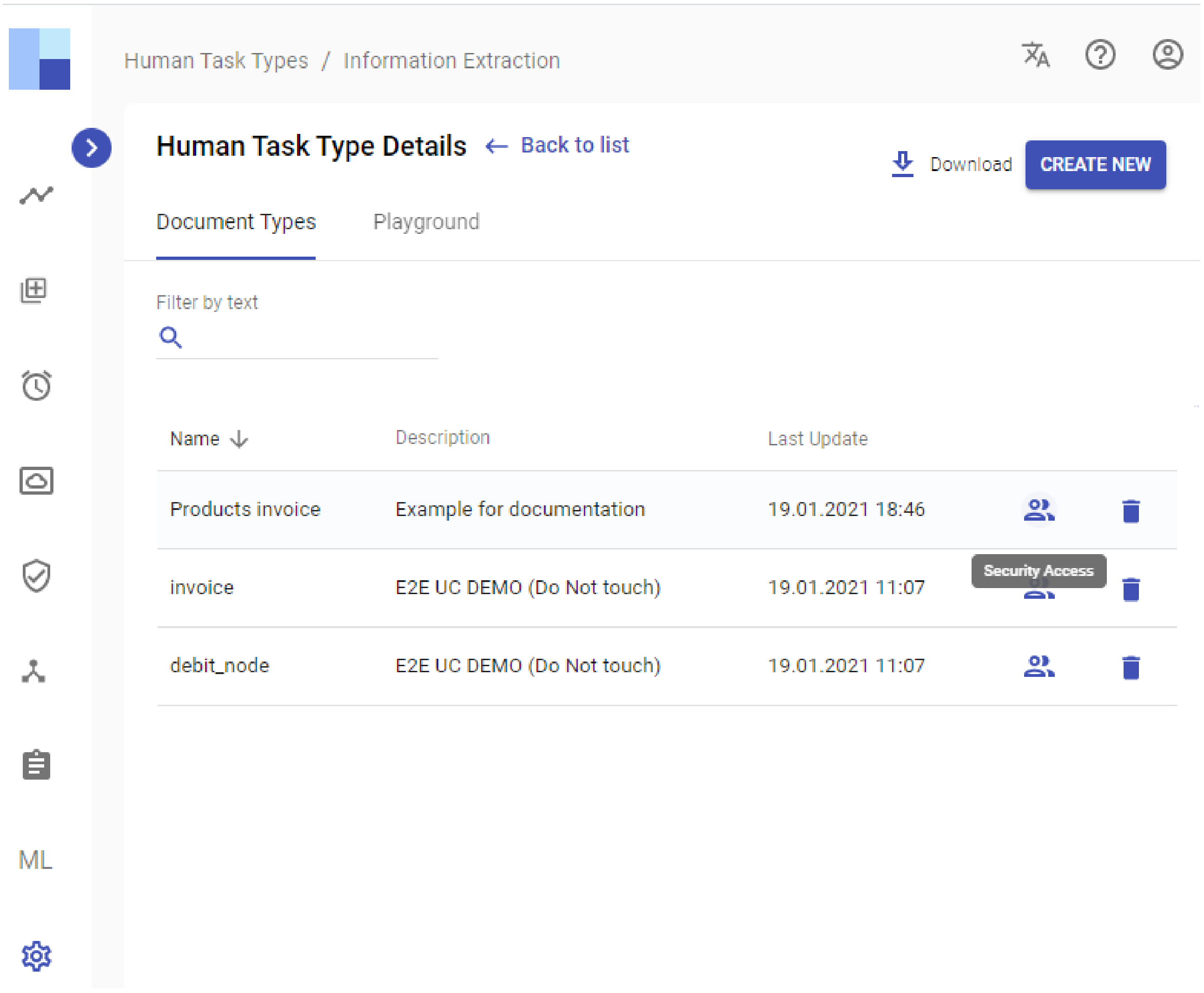
CONTRACT

19% + −

Access restriction for Document Type

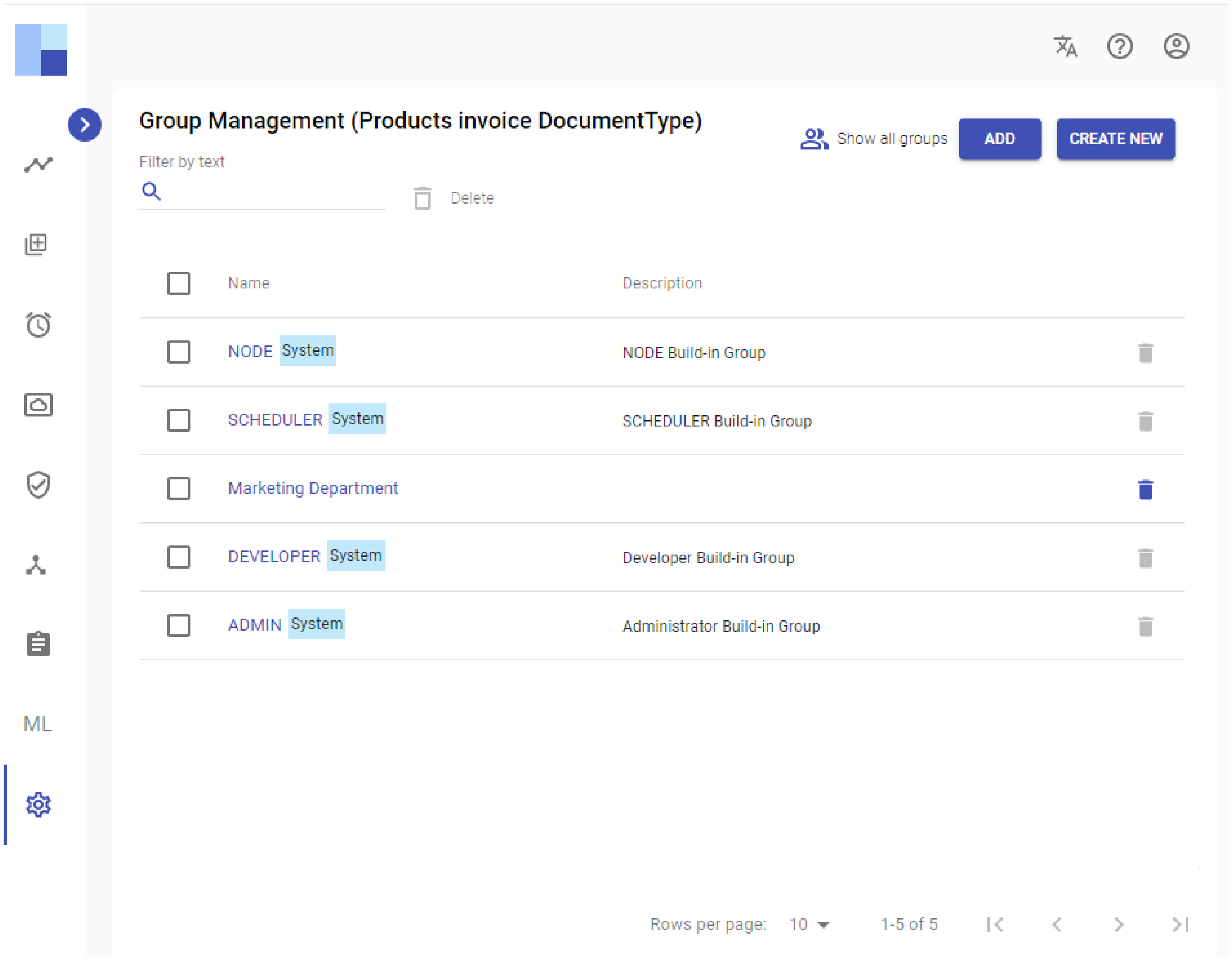
Each Document Type can be restricted to be visible only for some group of people. It's useful when you need to let different teams work only with document types they in charge of. E.g. your Insurance Department should have access to Insurance Claims document type while Marketing Department should be able to see only Advertising Invoices document type.

There's the special "Security Access" link on your Document Type to configure the list of groups which have access to that Document Type:



Name	Description	Last Update
Products invoice	Example for documentation	19.01.2021 18:46
invoice	E2E UC DEMO (Do Not touch)	19.01.2021 11:07
debit_node	E2E UC DEMO (Do Not touch)	19.01.2021 11:07

It opens the Group Management for that particular Document Type where you can manage accesses:



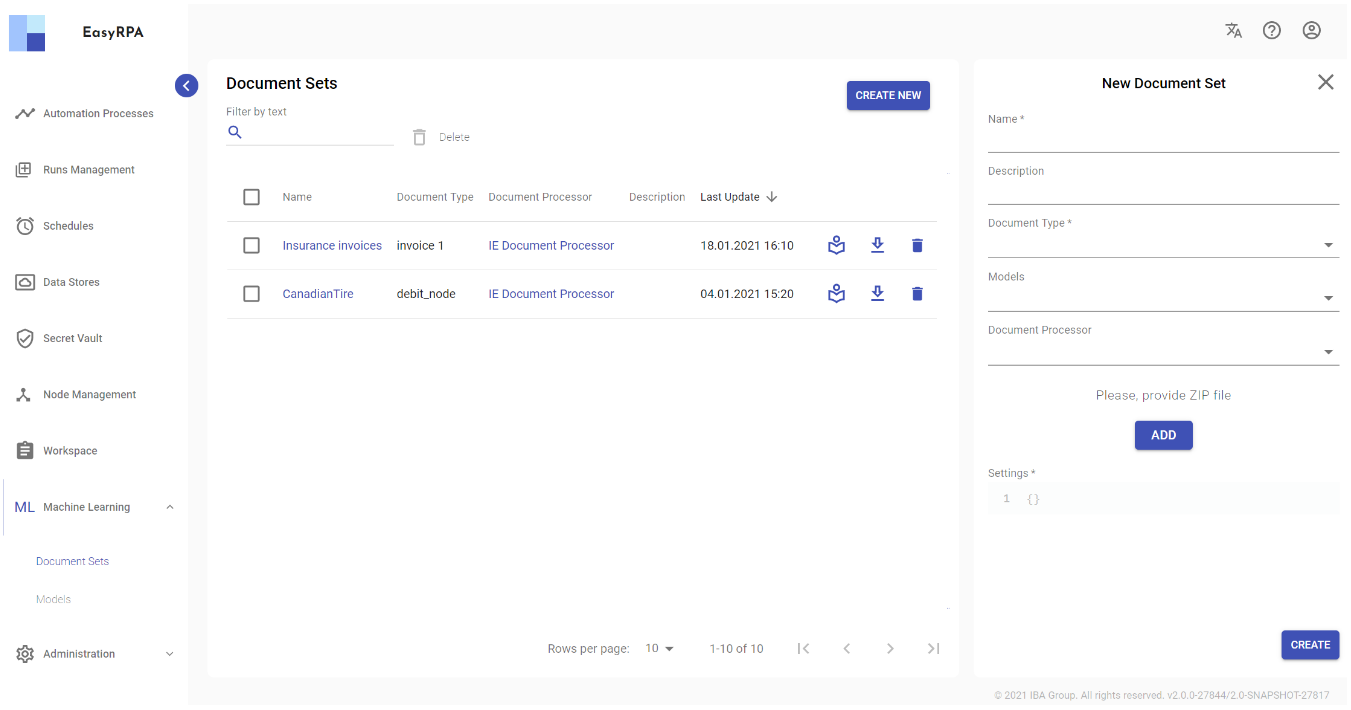
Group Management (Products invoice DocumentType)

Name	Description
NODE System	NODE Build-in Group
SCHEDULER System	SCHEDULER Build-in Group
Marketing Department	
DEVELOPER System	Developer Build-in Group
ADMIN System	Administrator Build-in Group

Rows per page: 10 1-5 of 5

(v. 2.2) Step 2. Collect and prepare training set (Classification)

To prepare the Training Set which is used to train new ML model, you need to create new **Document Set** entity. Click **Machine Learning** category **Document Sets** **Create New**:



	Name	Document Type	Document Processor	Description	Last Update
☐	Insurance invoices	invoice 1	IE Document Processor		18.01.2021 16:10
☐	CanadianTire	debit_node	IE Document Processor		04.01.2021 15:20

```
1  {}
```

© 2021 IBA Group. All rights reserved. v2.0.0-27844/2.0-SNAPSHOT-27817

Then training set preparation contains the following steps:

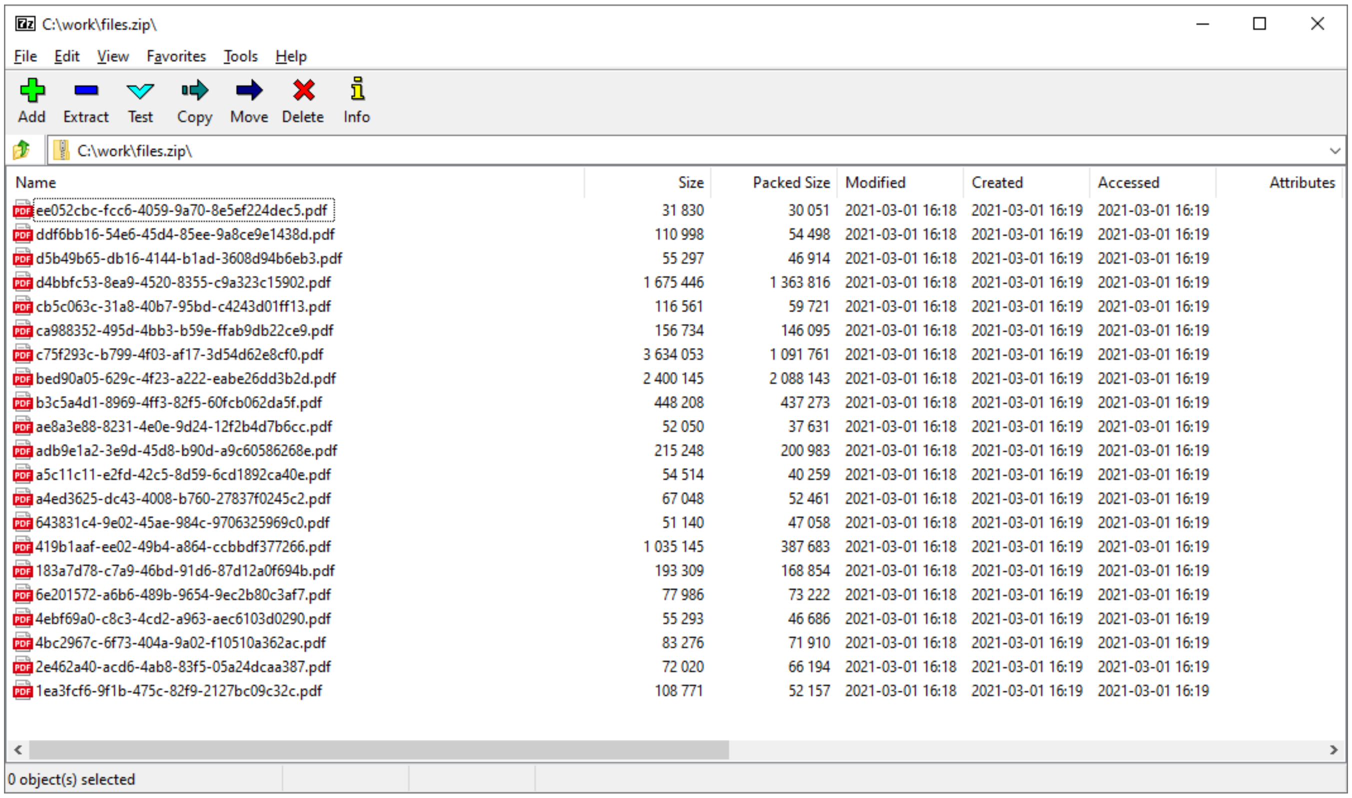
- Provide ZIP file
- Select Document Type
- Select Document Processor
- Generate Document Input
- Move documents to Human Task and tag them

Provide ZIP file

The main information of your training set is your documents. You need to put inside ZIP archive the set of your documents you want to use to train new ML model.

By default EasyRPA supports processing of **PDF** and **Images** documents to train a model.

Your archive should looks like the following:


C:\work\files.zip\

Name	Size	Packed Size	Modified	Created	Accessed	Attributes
ee052cbc-fcc6-4059-9a70-8e5ef224dec5.pdf	31 830	30 051	2021-03-01 16:18	2021-03-01 16:19	2021-03-01 16:19	
ddf6bb16-54e6-45d4-85ee-9a8ce9e1438d.pdf	110 998	54 498	2021-03-01 16:18	2021-03-01 16:19	2021-03-01 16:19	
d5b49b65-db16-4144-b1ad-3608d94b6eb3.pdf	55 297	46 914	2021-03-01 16:18	2021-03-01 16:19	2021-03-01 16:19	
d4bbfc53-8ea9-4520-8355-c9a323c15902.pdf	1 675 446	1 363 816	2021-03-01 16:18	2021-03-01 16:19	2021-03-01 16:19	
cb5c063c-31a8-40b7-95bd-c4243d01ff13.pdf	116 561	59 721	2021-03-01 16:18	2021-03-01 16:19	2021-03-01 16:19	
ca988352-495d-4bb3-b59e-ffab9db22ce9.pdf	156 734	146 095	2021-03-01 16:18	2021-03-01 16:19	2021-03-01 16:19	
c75f293c-b799-4f03-af17-3d54d62e8cf0.pdf	3 634 053	1 091 761	2021-03-01 16:18	2021-03-01 16:19	2021-03-01 16:19	
bed90a05-629c-4f23-a222-eabe26dd3b2d.pdf	2 400 145	2 088 143	2021-03-01 16:18	2021-03-01 16:19	2021-03-01 16:19	
b3c5a4d1-8969-4ff3-82f5-60fcb062da5f.pdf	448 208	437 273	2021-03-01 16:18	2021-03-01 16:19	2021-03-01 16:19	
ae8a3e88-8231-4e0e-9d24-12f2b4d7b6cc.pdf	52 050	37 631	2021-03-01 16:18	2021-03-01 16:19	2021-03-01 16:19	
adb9e1a2-3e9d-45d8-b90d-a9c60586268e.pdf	215 248	200 983	2021-03-01 16:18	2021-03-01 16:19	2021-03-01 16:19	
a5c11c11-e2fd-42c5-8d59-6cd1892ca40e.pdf	54 514	40 259	2021-03-01 16:18	2021-03-01 16:19	2021-03-01 16:19	
a4ed3625-dc43-4008-b760-27837f0245c2.pdf	67 048	52 461	2021-03-01 16:18	2021-03-01 16:19	2021-03-01 16:19	
643831c4-9e02-45ae-984c-9706325969c0.pdf	51 140	47 058	2021-03-01 16:18	2021-03-01 16:19	2021-03-01 16:19	
419b1aaf-ee02-49b4-a864-ccbbdf377266.pdf	1 035 145	387 683	2021-03-01 16:18	2021-03-01 16:19	2021-03-01 16:19	
183a7d78-c7a9-46bd-91d6-87d12a0f694b.pdf	193 309	168 854	2021-03-01 16:18	2021-03-01 16:19	2021-03-01 16:19	
6e201572-a6b6-489b-9654-9ec2b80c3af7.pdf	77 986	73 222	2021-03-01 16:18	2021-03-01 16:19	2021-03-01 16:19	
4ebf69a0-c8c3-4cd2-a963-aec6103d0290.pdf	55 293	46 686	2021-03-01 16:18	2021-03-01 16:19	2021-03-01 16:19	
4bc2967c-6f73-404a-9a02-f10510a362ac.pdf	83 276	71 910	2021-03-01 16:18	2021-03-01 16:19	2021-03-01 16:19	
2e462a40-acd6-4ab8-83f5-05a24dcaa387.pdf	72 020	66 194	2021-03-01 16:18	2021-03-01 16:19	2021-03-01 16:19	
1ea3fcf6-9f1b-475c-82f9-2127bc09c32c.pdf	108 771	52 157	2021-03-01 16:18	2021-03-01 16:19	2021-03-01 16:19	

0 object(s) selected

Select Document Type

Select Document Type created on the previous step. This Document Type contains information of how to display your documents on Human Task for classification and which classes should be considered for Machine Learning training.

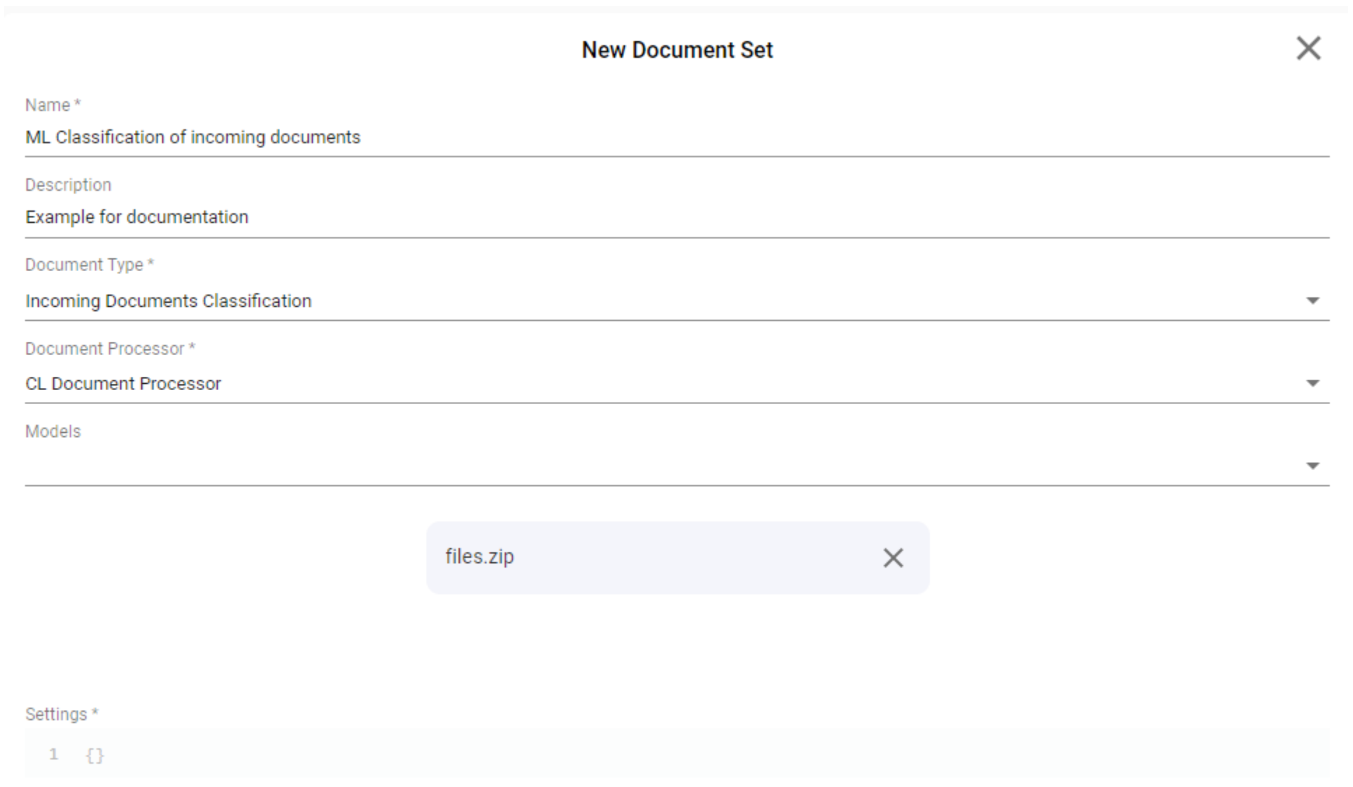
Select Document Processor

Document Processor is a special process which includes preparation steps to display your documents on the Human Task.

To process PDFs and Images for Classification, EasyRPA provides "**CL Document Processor**" which should be selected in our case. This processor includes:

- converting PDFs into images format
- images improvements using ImageMagick scripts
- sending images to OCR
- sending documents with OCR response to Human Task
- process Human Task response to convert results into Machine Learning input format

Finally the new Document Set configuration for Classification should looks like below:



New Document Set

Name *
ML Classification of incoming documents

Description
Example for documentation

Document Type *
Incoming Documents Classification

Document Processor *
CL Document Processor

Models

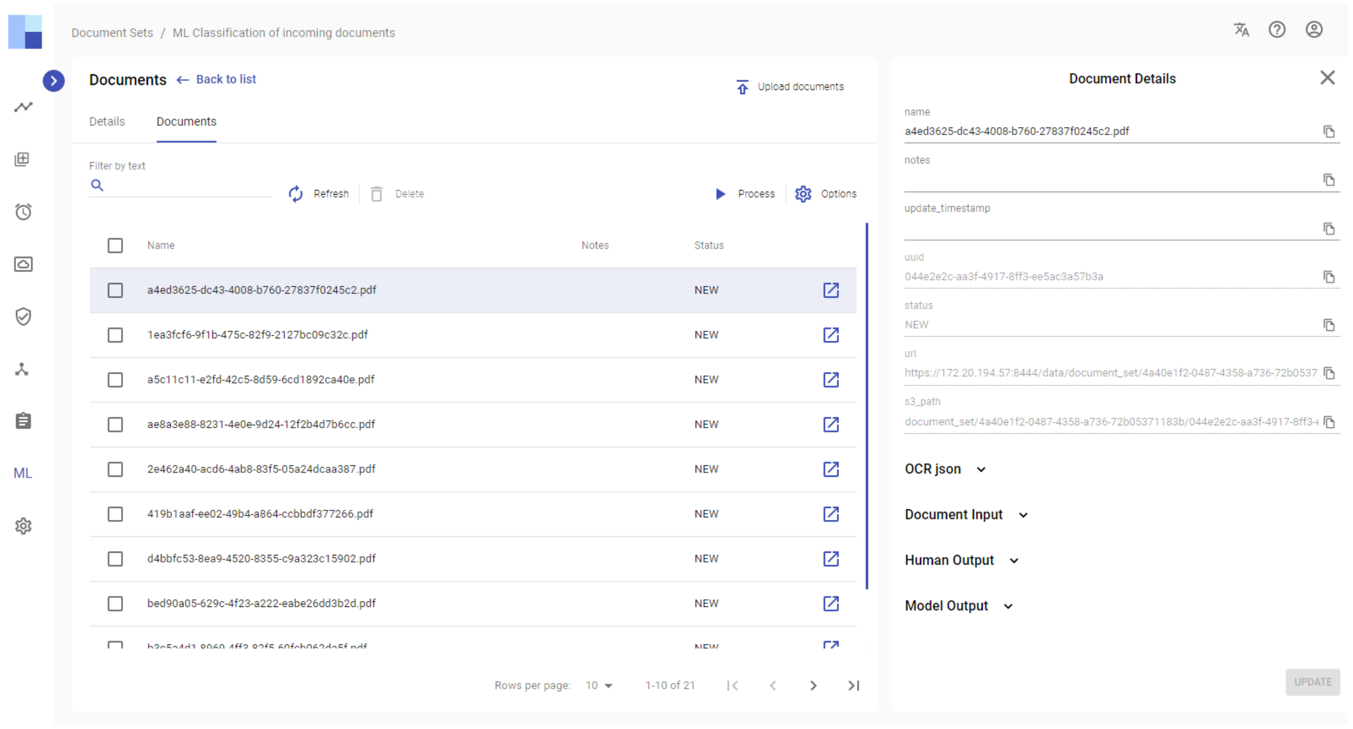
files.zip

Settings *
```
{}
```

After you provided ZIP archive, selected Document Type and Document Processor, you can create document set.

All documents from your Document Set will be uploaded to File Storage.

If you open created Document Set, you can see the details of each document inside:



Name	Notes	Status
a4ed3625-dc43-4008-b760-27837f0245c2.pdf		NEW
1ea3fcf6-9f1b-475c-82f9-2127bc09c32c.pdf		NEW
a5c11c11-e2fd-42c5-8d59-6cd1892ca40e.pdf		NEW
ae8a3e88-8231-4e0e-9d24-12f2b4d7b6cc.pdf		NEW
2e462a40-acd6-4ab8-83f5-05a24dcaa387.pdf		NEW
419b1aaf-ee02-49b4-a864-ccbbdf377266.pdf		NEW
d4bbfc53-8ea9-4520-8355-c9a323c15902.pdf		NEW
bed90a05-629c-4f23-a222-eabe26dd3b2d.pdf		NEW
b3c5a4d1-8060-4ff3-82f5-60fcb062da5f.pdf		NEW

Document Details

name
a4ed3625-dc43-4008-b760-27837f0245c2.pdf

notes

update_timestamp

uuid
044e2e2c-aa3f-4917-8ff3-ee5ac3a57b3a

status
NEW

url
https://172.20.194.57:8444/data/document_set/4a40e1f2-0487-4358-a736-72b0537

s3_path
document_set/4a40e1f2-0487-4358-a736-72b05371183b/044e2e2c-aa3f-4917-8ff3-

OCR json

Document Input

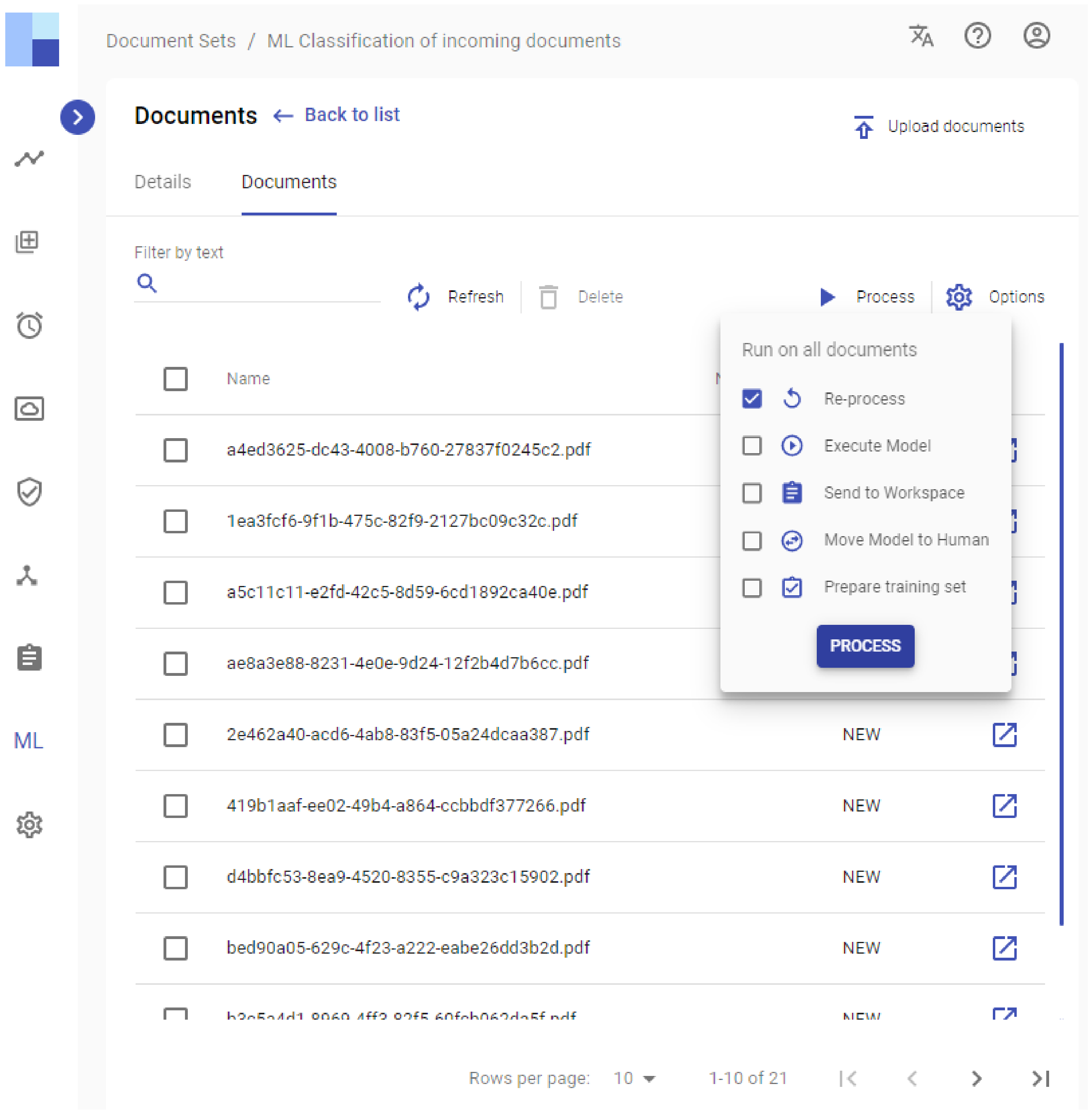
Human Output

Model Output

As this Document Set is new, **Document Input**, **Human Output** and **Model Output** details are empty.

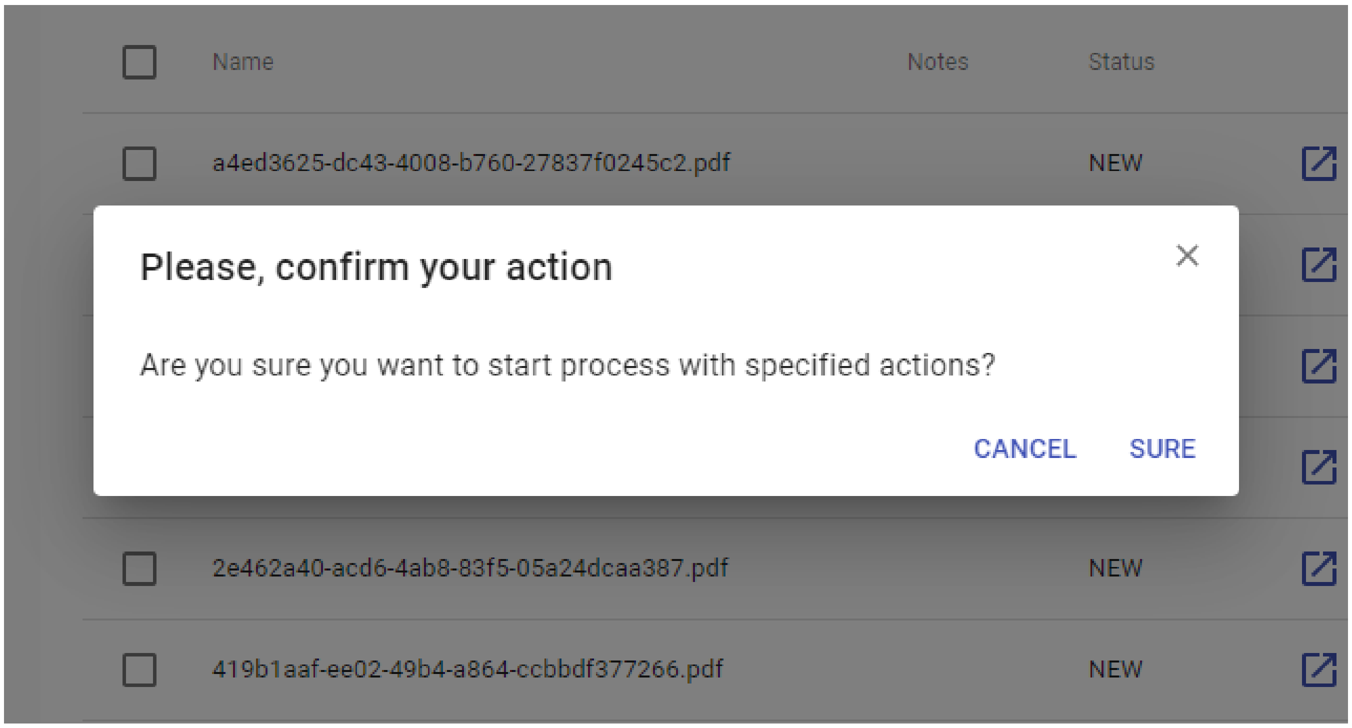
Generate Document Input

Document Input is an input data which comes as an input for Human Tasks. As mentioned before, Document Processor is responsible for preparation of that input. To trigger Document Processor and to prepare Document Input, you should **select "Re-process" action** from the action menu and then confirm the in appeared confirmation dialog:



	Name		
☐	a4ed3625-dc43-4008-b760-27837f0245c2.pdf		
☐	1ea3fcf6-9f1b-475c-82f9-2127bc09c32c.pdf		
☐	a5c11c11-e2fd-42c5-8d59-6cd1892ca40e.pdf		
☐	ae8a3e88-8231-4e0e-9d24-12f2b4d7b6cc.pdf		
☐	2e462a40-acd6-4ab8-83f5-05a24dcaa387.pdf	NEW	
☐	419b1aaf-ee02-49b4-a864-ccbbdf377266.pdf	NEW	
☐	d4bbfc53-8ea9-4520-8355-c9a323c15902.pdf	NEW	
☐	bed90a05-629c-4f23-a222-eabe26dd3b2d.pdf	NEW	

Rows per page: 10 1-10 of 21

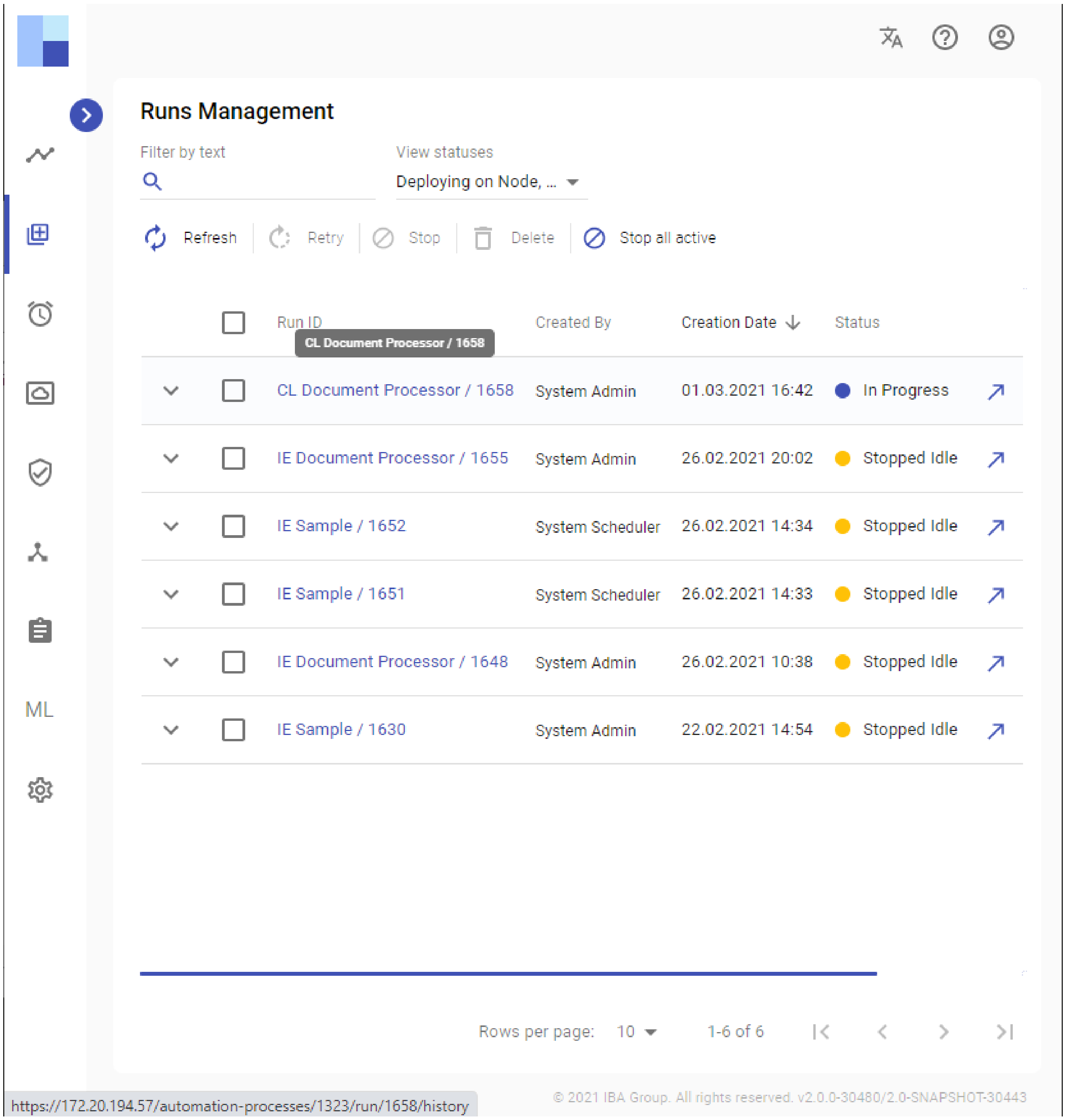


	Name	Notes	Status
☐	a4ed3625-dc43-4008-b760-27837f0245c2.pdf		NEW
☐	2e462a40-acd6-4ab8-83f5-05a24dcaa387.pdf		NEW
☐	419b1aaf-ee02-49b4-a864-ccbbdf377266.pdf		NEW

Please, confirm your action

Are you sure you want to start process with specified actions?

Document Processor will start working. You can monitor it in Runs Management:

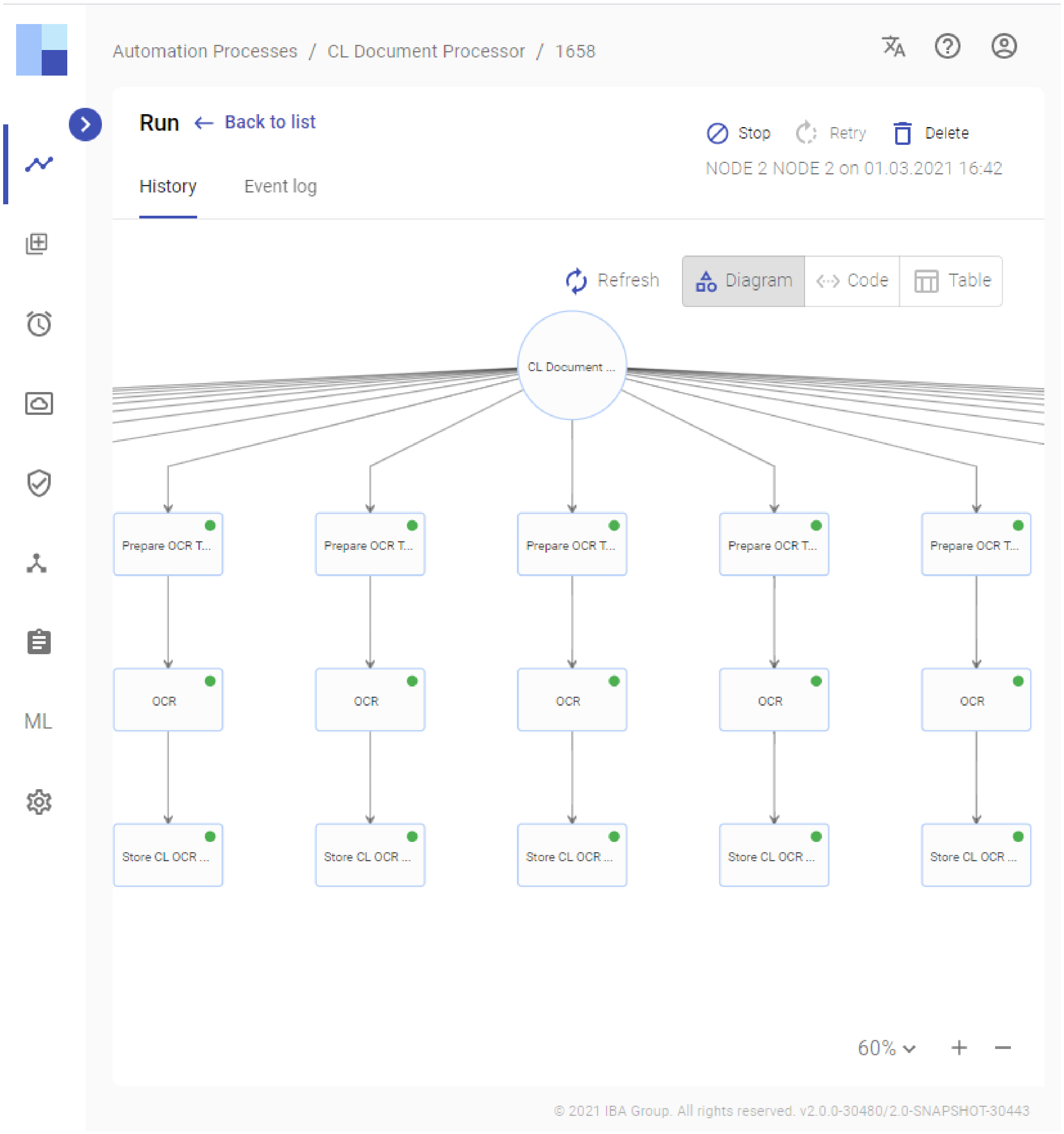


	Run ID	Created By	Creation Date	Status
	CL Document Processor / 1658	System Admin	01.03.2021 16:42	● In Progress
	IE Document Processor / 1655	System Admin	26.02.2021 20:02	● Stopped Idle
	IE Sample / 1652	System Scheduler	26.02.2021 14:34	● Stopped Idle
	IE Sample / 1651	System Scheduler	26.02.2021 14:33	● Stopped Idle
	IE Document Processor / 1648	System Admin	26.02.2021 10:38	● Stopped Idle
	IE Sample / 1630	System Admin	22.02.2021 14:54	● Stopped Idle

Rows per page: 10 1-6 of 6

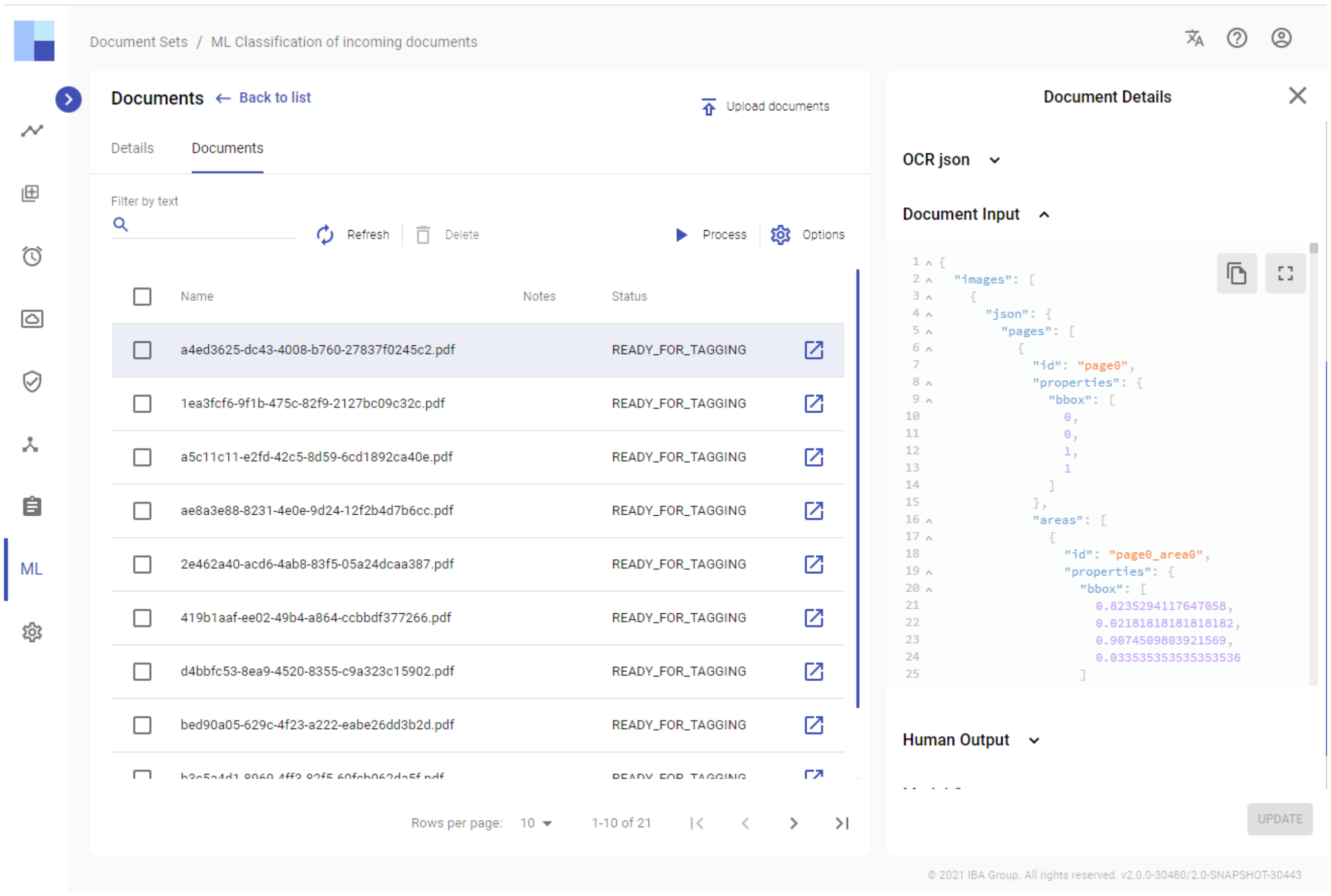
https://172.20.194.57/automation-processes/1323/run/1658/history

© 2021 IBA Group. All rights reserved. v2.0.0-30480/2.0-SNAPSHOT-30443


Automation Processes / CL Document Processor / 1658
Run
← Back to list
Stop
Retry
Delete
NODE 2 NODE 2 on 01.03.2021 16:42
History
Event log
Refresh
Diagram
Code
Table
CL Document ...
Prepare OCR T...
OCR
Store CL OCR ...
60%
ML
© 2021 IBA Group. All rights reserved. v2.0.0-30480/2.0-SNAPSHOT-30443

Wait until the status of all documents becomes "**READY_FOR_TAGGING**"

As the result you will see that "**Document input**" details are generated where you will find the input data from Human Task. This data contains the result of OCR in an appropriate format:


Document Sets / ML Classification of incoming documents

Documents ← Back to list

Upload documents

Details Documents

Filter by text

Refresh Delete Process Options

Name	Notes	Status
a4ed3625-dc43-4008-b760-27837f0245c2.pdf		READY_FOR_TAGGING
1ea3fcf6-9f1b-475c-82f9-2127bc09c32c.pdf		READY_FOR_TAGGING
a5c11c11-e2fd-42c5-8d59-6cd1892ca40e.pdf		READY_FOR_TAGGING
ae8a3e88-8231-4e0e-9d24-12f2b4d7b6cc.pdf		READY_FOR_TAGGING
2e462a40-acd6-4ab8-83f5-05a24dcaa387.pdf		READY_FOR_TAGGING
419b1aaf-ee02-49b4-a864-ccbbdf377266.pdf		READY_FOR_TAGGING
d4bbfc53-8ea9-4520-8355-c9a323c15902.pdf		READY_FOR_TAGGING
bed90a05-629c-4f23-a222-eabe26dd3b2d.pdf		READY_FOR_TAGGING

Rows per page: 10 1-10 of 21

Document Details

OCR json

Document Input

```
{
  "images": [
    {
      "json": {
        "pages": [
          {
            "id": "page0",
            "properties": {
              "bbox": [
                0,
                0,
                1,
                1
              ]
            },
            "areas": [
              {
                "id": "page0_area0",
                "properties": {
                  "bbox": [
                    0.8235294117647058,
                    0.02181818181818182,
                    0.9074509803921569,
                    0.033535353535353536
                  ]
```

Human Output

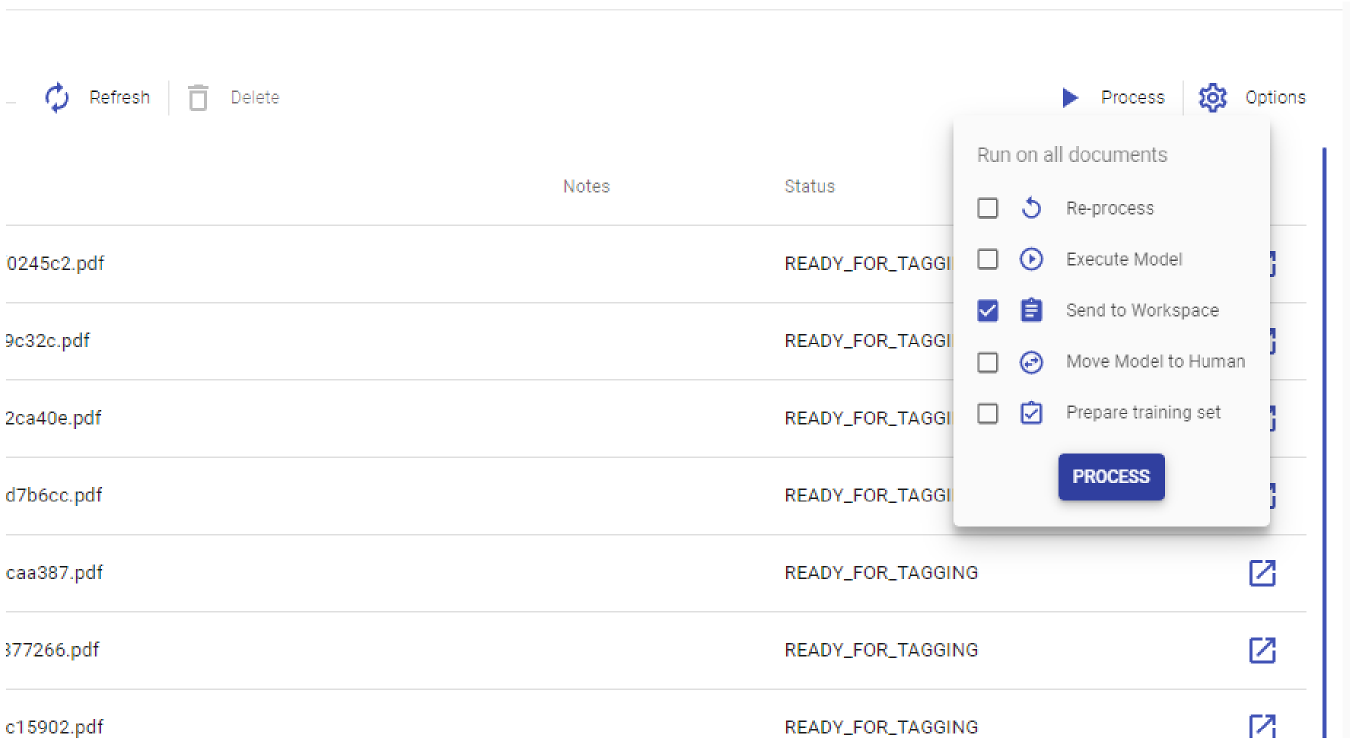
UPDATE

© 2021 IBA Group. All rights reserved. v2.0.0-30480/2.0-SNAPSHOT-30443

Move documents to Human Task and tag them

The next step we need to classify the documents. This result of classification will be used by ML algorithms to train a model.

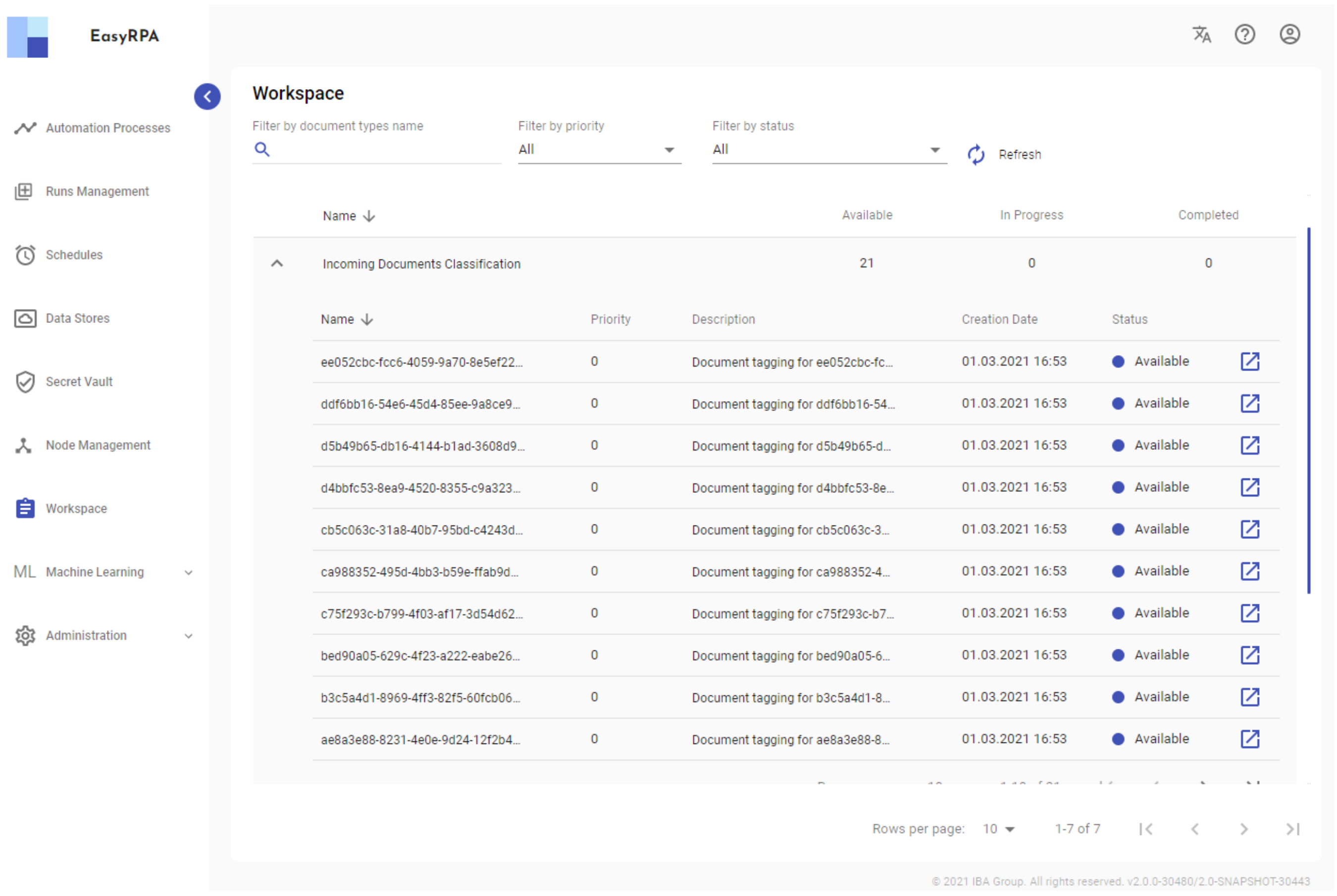
To move your Document Set to Workspace and to let workers to tag them via Human Task you need to select "**Send to Workspace**" action:



	Notes	Status
0245c2.pdf		READY_FOR_TAGGI
9c32c.pdf		READY_FOR_TAGGI
2ca40e.pdf		READY_FOR_TAGGI
d7b6cc.pdf		READY_FOR_TAGGI
caa387.pdf		READY_FOR_TAGGING
377266.pdf		READY_FOR_TAGGING
c15902.pdf		READY_FOR_TAGGING

Send to Workspace [✓]

Documents should be appeared in Workspace under corresponding Document Type. You can click button to start tagging it:



Workspace

Filter by document types name

Filter by priority: All

Filter by status: All

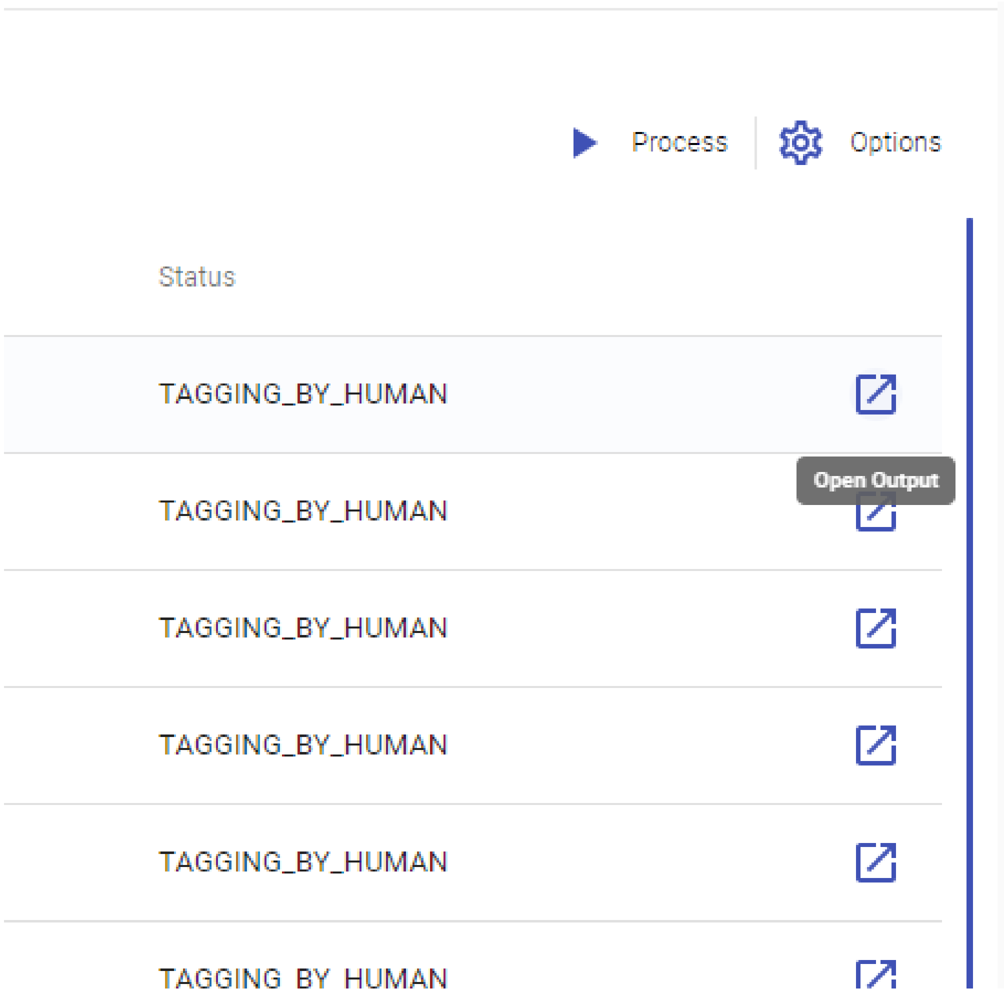
Name	Available	In Progress	Completed
Incoming Documents Classification	21	0	0

Name	Priority	Description	Creation Date	Status
ee052cbc-fcc6-4059-9a70-8e5ef22...	0	Document tagging for ee052cbc-fc...	01.03.2021 16:53	● Available
ddf6bb16-54e6-45d4-85ee-9a8ce9...	0	Document tagging for ddf6bb16-54...	01.03.2021 16:53	● Available
d5b49b65-db16-4144-b1ad-3608d9...	0	Document tagging for d5b49b65-d...	01.03.2021 16:53	● Available
d4bbfc53-8ea9-4520-8355-c9a323...	0	Document tagging for d4bbfc53-8e...	01.03.2021 16:53	● Available
cb5c063c-31a8-40b7-95bd-c4243d...	0	Document tagging for cb5c063c-3...	01.03.2021 16:53	● Available
ca988352-495d-4bb3-b59e-ffab9d...	0	Document tagging for ca988352-4...	01.03.2021 16:53	● Available
c75f293c-b799-4f03-af17-3d54d62...	0	Document tagging for c75f293c-b7...	01.03.2021 16:53	● Available
bed90a05-629c-4f23-a222-eabe26...	0	Document tagging for bed90a05-6...	01.03.2021 16:53	● Available
b3c5a4d1-8969-4ff3-82f5-60fcb06...	0	Document tagging for b3c5a4d1-8...	01.03.2021 16:53	● Available
ae8a3e88-8231-4e0e-9d24-12f2b4...	0	Document tagging for ae8a3e88-8...	01.03.2021 16:53	● Available

Rows per page: 10 1-7 of 7

© 2021 IBA Group. All rights reserved. v2.0.0-30480/2.0-SNAPSHOT-30443

** There's also an option to classify document without sending it to the Workspace. So you can click "Open Output" button next to the status label inside you Document Set details:*



Status

TAGGING_BY_HUMAN

TAGGING_BY_HUMAN

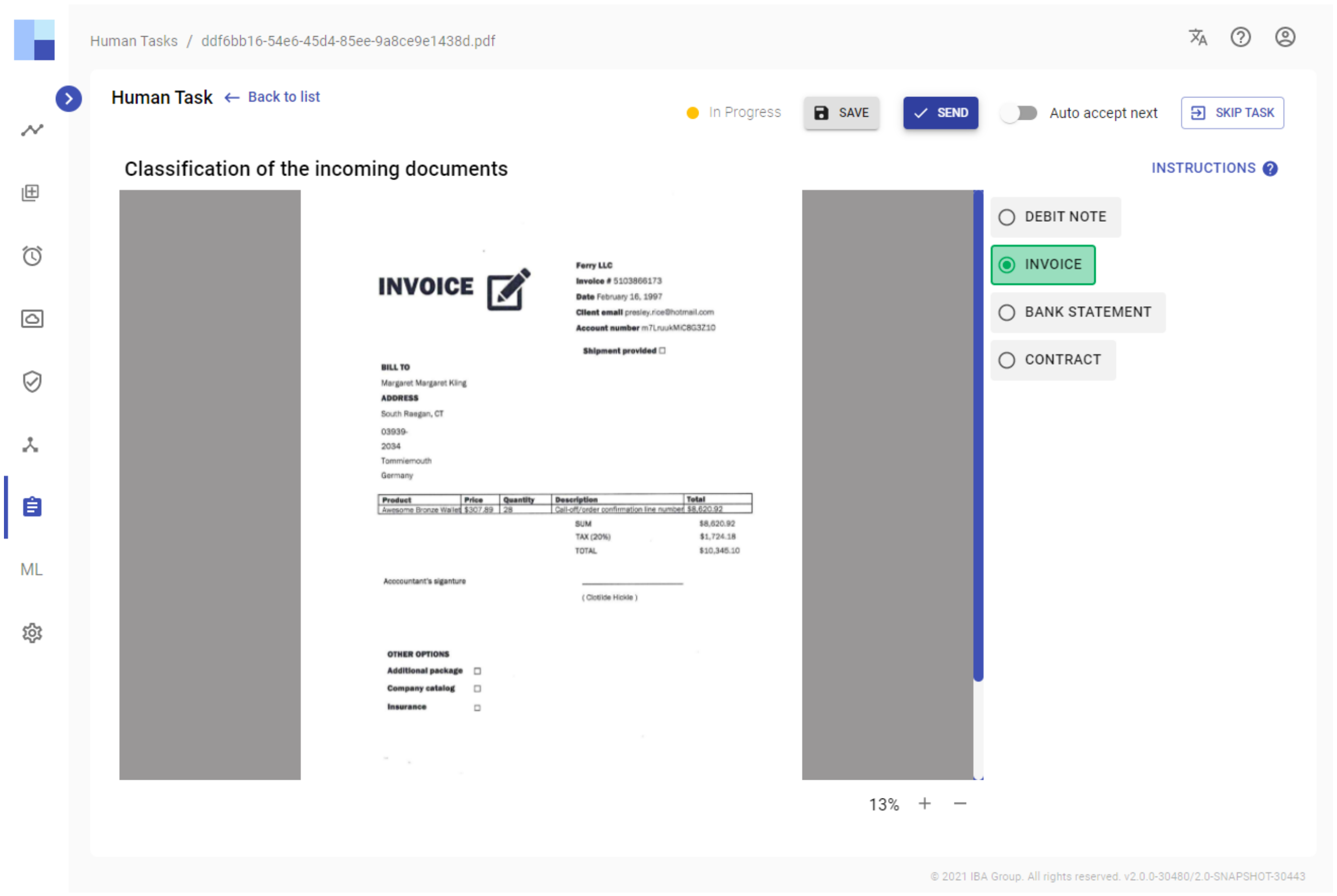
TAGGING_BY_HUMAN

TAGGING_BY_HUMAN

TAGGING_BY_HUMAN

TAGGING_BY_HUMAN

You can start document classification:



Classification of the incoming documents

DEBIT NOTE
[✓ INVOICE]
BANK STATEMENT
CONTRACT

INVOICE

Ferry LLC
Invoice # 5103866173
Date February 16, 1997
Client email presley.rice@hotmail.com
Account number m7LruukMiC8G3Z10

Shipment provided ☐

BILL TO
Margaret Margaret Kling
ADDRESS
South Raegan, CT
03939-
2034
Tommiemouth
Germany

Product	Price	Quantity	Description	Total
Awesome Bronze Wallet	$307.89	28	Call-off/order confirmation line number	$8,620.92

SUM	$8,620.92
TAX (20%)	$1,724.18
TOTAL	$10,346.10

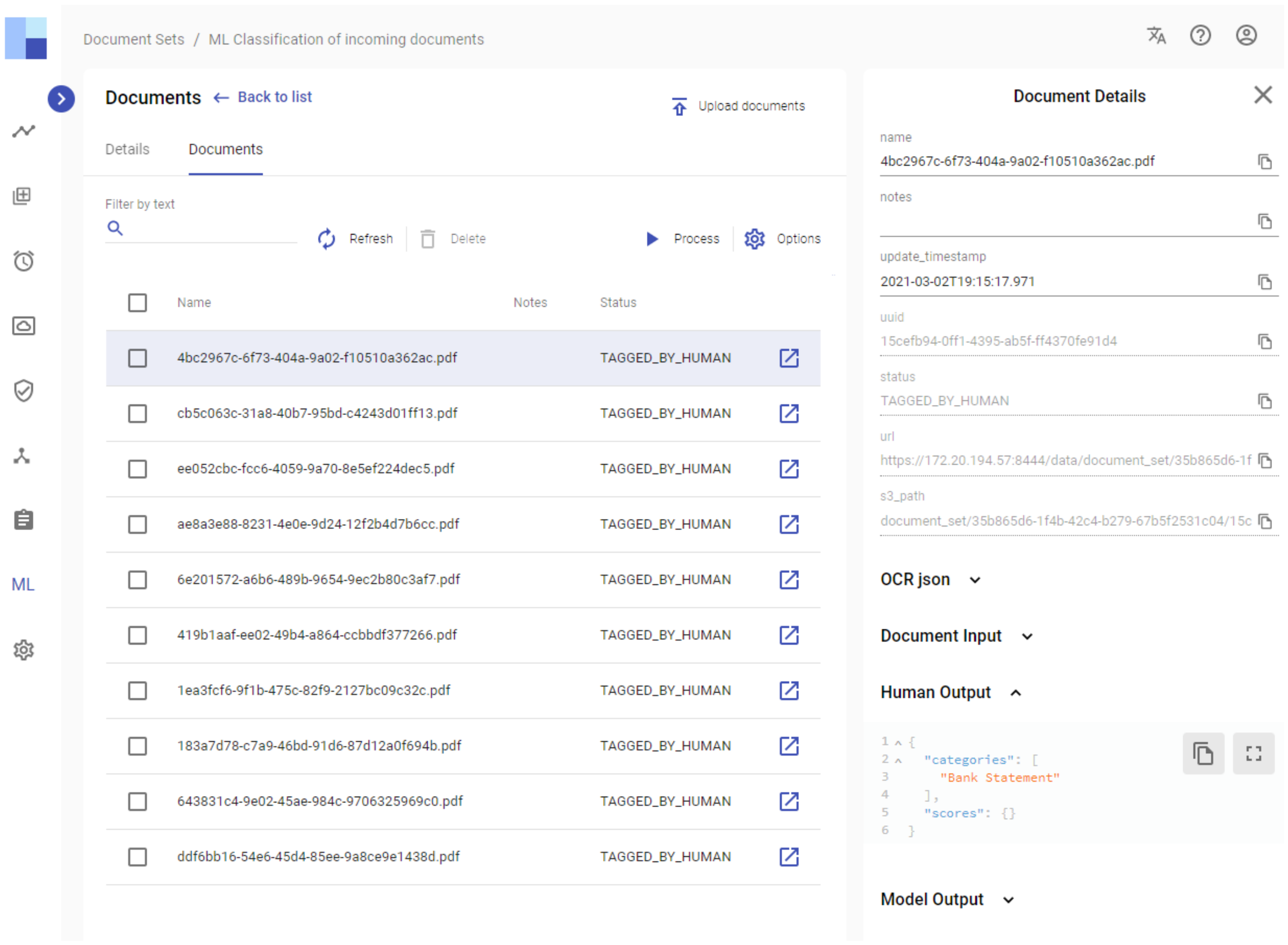
Acocountant's siganture

(Clotilde Hickle)

OTHER OPTIONS
Additional package ☐
Company catalog ☐
Insurance ☐

© 2021 IBA Group. All rights reserved. v2.0.0-30480/2.0-SNAPSHOT-30443

After document is tagged, "**Human Output**" details are generated for that particular document:


Document Sets / ML Classification of incoming documents

Documents ← Back to list

Upload documents

Details | Documents

Filter by text

Refresh | Delete | Process | Options

	Name	Notes	Status
☐	4bc2967c-6f73-404a-9a02-f10510a362ac.pdf		TAGGED_BY_HUMAN
☐	cb5c063c-31a8-40b7-95bd-c4243d01ff13.pdf		TAGGED_BY_HUMAN
☐	ee052cbc-fcc6-4059-9a70-8e5ef224dec5.pdf		TAGGED_BY_HUMAN
☐	ae8a3e88-8231-4e0e-9d24-12f2b4d7b6cc.pdf		TAGGED_BY_HUMAN
☐	6e201572-a6b6-489b-9654-9ec2b80c3af7.pdf		TAGGED_BY_HUMAN
☐	419b1aaf-ee02-49b4-a864-ccbbdf377266.pdf		TAGGED_BY_HUMAN
☐	1ea3fcf6-9f1b-475c-82f9-2127bc09c32c.pdf		TAGGED_BY_HUMAN
☐	183a7d78-c7a9-46bd-91d6-87d12a0f694b.pdf		TAGGED_BY_HUMAN
☐	643831c4-9e02-45ae-984c-9706325969c0.pdf		TAGGED_BY_HUMAN
☐	ddf6bb16-54e6-45d4-85ee-9a8ce9e1438d.pdf		TAGGED_BY_HUMAN

Document Details

name: 4bc2967c-6f73-404a-9a02-f10510a362ac.pdf

notes:

update_timestamp: 2021-03-02T19:15:17.971

uuid: 15cefb94-0ff1-4395-ab5f-ff4370fe91d4

status: TAGGED_BY_HUMAN

url: https://172.20.194.57:8444/data/document_set/35b865d6-1f

s3_path: document_set/35b865d6-1f4b-42c4-b279-67b5f2531c04/15c

OCR json

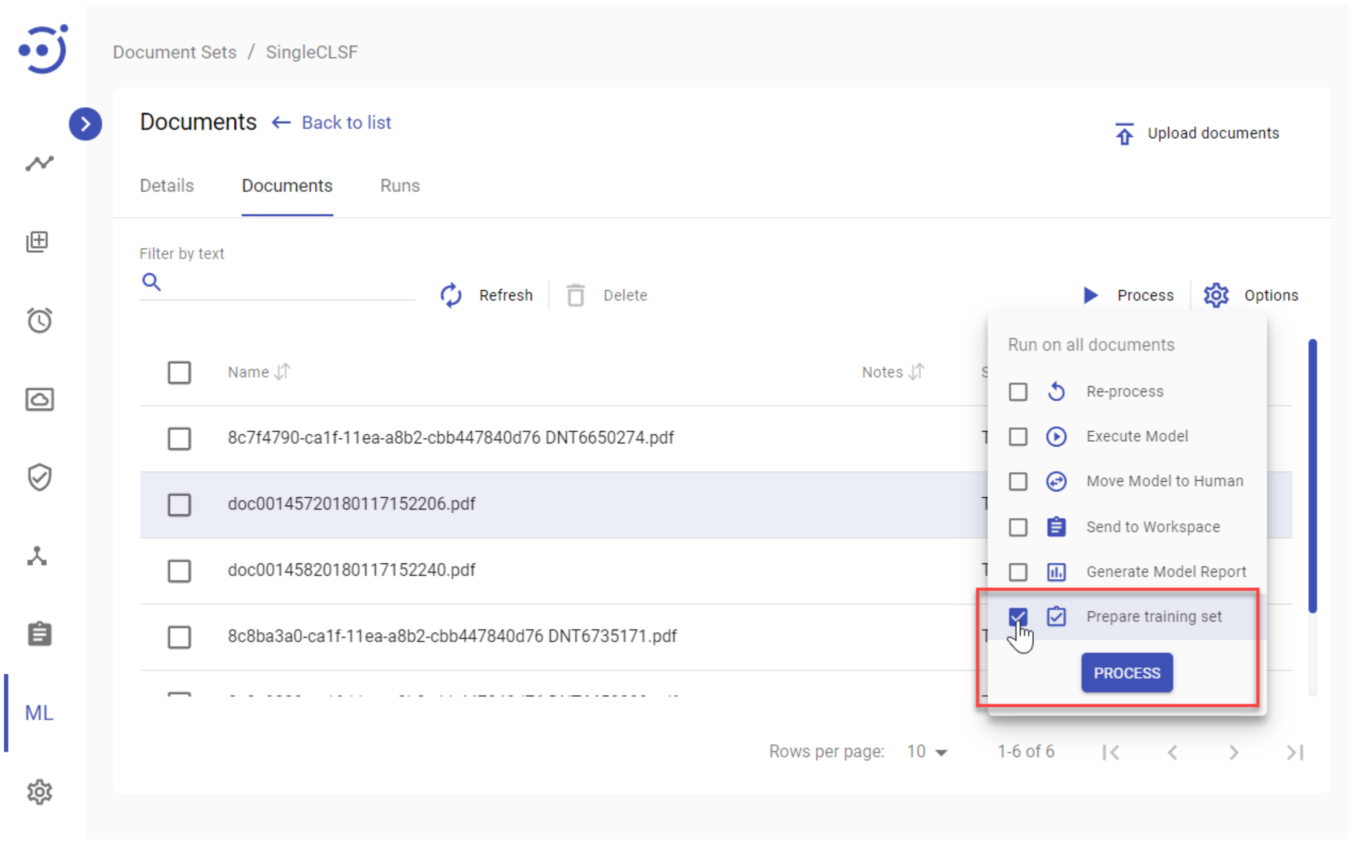
Document Input

Human Output

```
{
  "categories": [
    "Bank Statement"
  ],
  "scores": {}
}
```

Model Output

To finish Training set preparation click **Process** **Prepare training set** and confirm your action with **Process** button:


Document Sets / SingleCLSF

Documents ← Back to list

Upload documents

Details Documents Runs

Filter by text

Refresh Delete Process Options

Name	Notes
8c7f4790-ca1f-11ea-a8b2-cbb447840d76 DNT6650274.pdf	
doc00145720180117152206.pdf	
doc00145820180117152240.pdf	
8c8ba3a0-ca1f-11ea-a8b2-cbb447840d76 DNT6735171.pdf	

Run on all documents
- [ ] Re-process
- [ ] Execute Model
- [ ] Move Model to Human
- [ ] Send to Workspace
- [ ] Generate Model Report
- [✓] Prepare training set

PROCESS

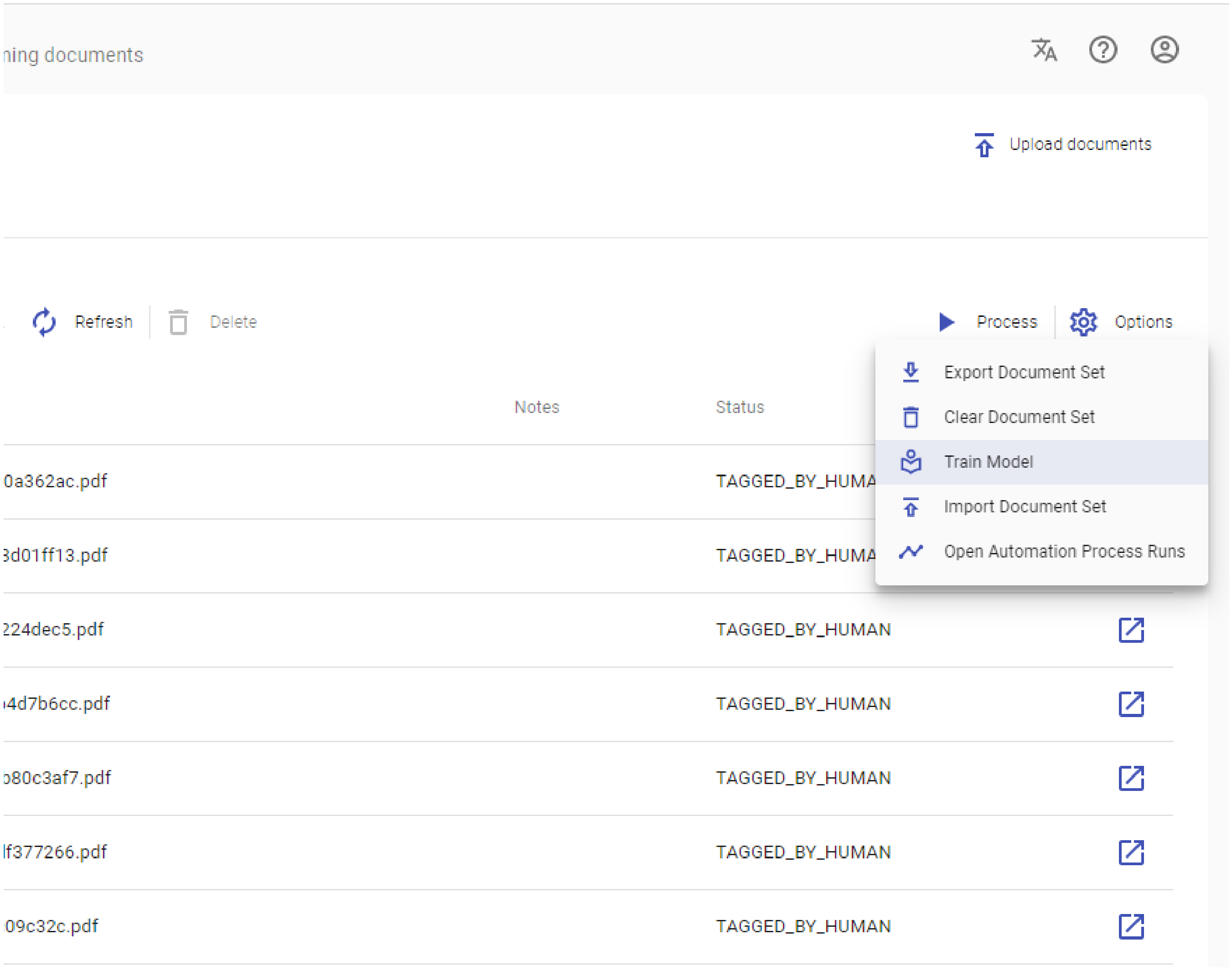
Rows per page: 10 1-6 of 6

After completing these actions you can proceed to model training.

(v. 2.2) Step 3. Train ML model (Classification)

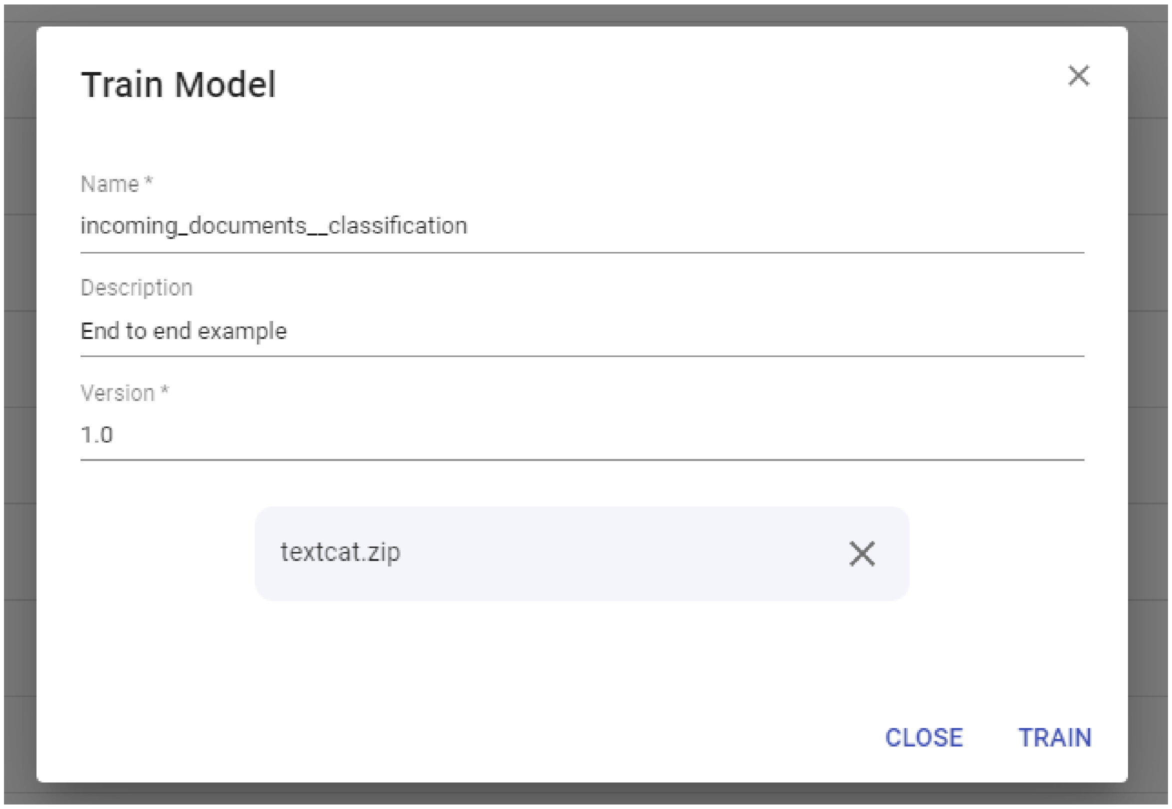
After you prepared training set and tag all the documents, you can train your ML model.

Inside Document Set select **Options Train Model**



	Notes	Status
0a362ac.pdf		TAGGED_BY_HUMA
3d01ff13.pdf		TAGGED_BY_HUMA
224dec5.pdf		TAGGED_BY_HUMAN
4d7b6cc.pdf		TAGGED_BY_HUMAN
080c3af7.pdf		TAGGED_BY_HUMAN
lf377266.pdf		TAGGED_BY_HUMAN
09c32c.pdf		TAGGED_BY_HUMAN

Following dialog appears:


Train Model

Name *
incoming_documents__classification

Description
End to end example

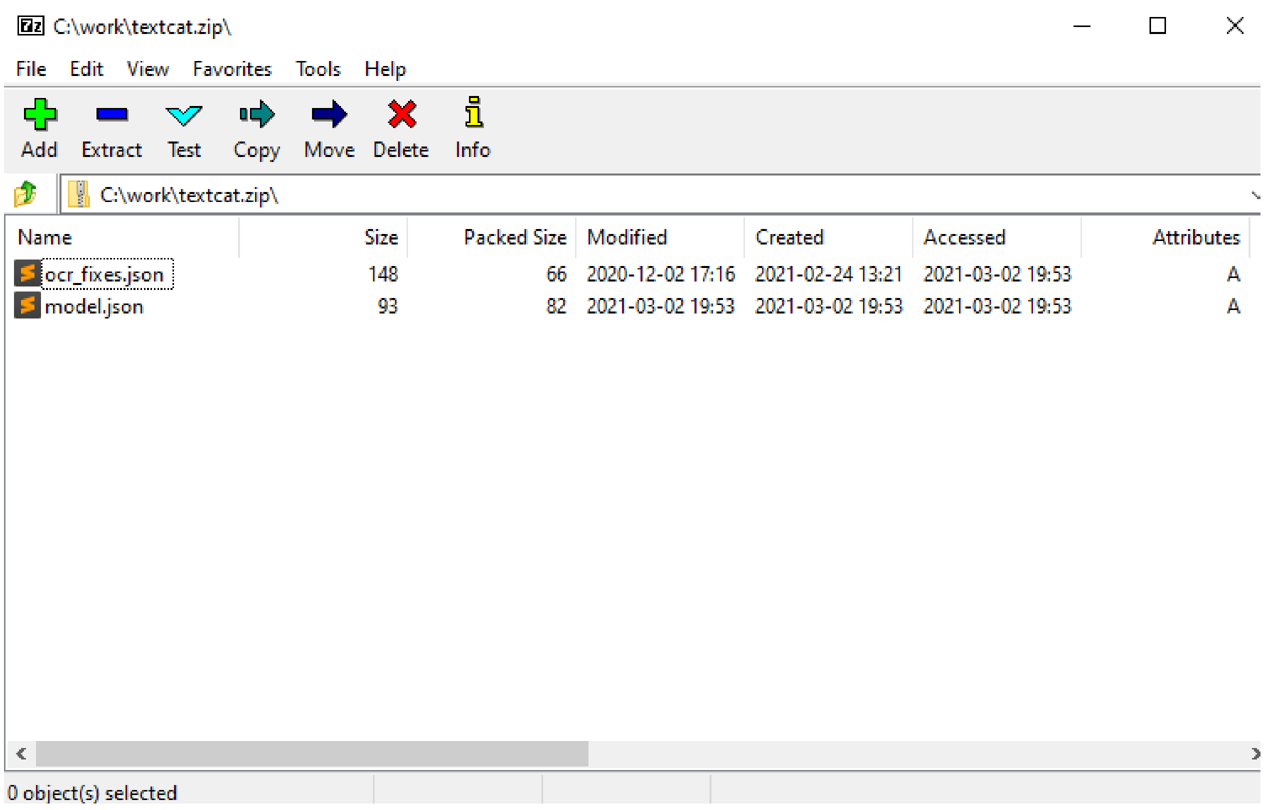
Version *
1.0

textcat.zip

You need to specify:

- Model name
- Model description
- Model version
- Attach ZIP file with configurations.

ZIP file with configurations must contains **model.json**



Name	Size	Packed Size	Modified	Created	Accessed	Attributes
ocr_fixes.json	148	66	2020-12-02 17:16	2021-02-24 13:21	2021-03-02 19:53	A
model.json	93	82	2021-03-02 19:53	2021-03-02 19:53	2021-03-02 19:53	A

model.json configuration file

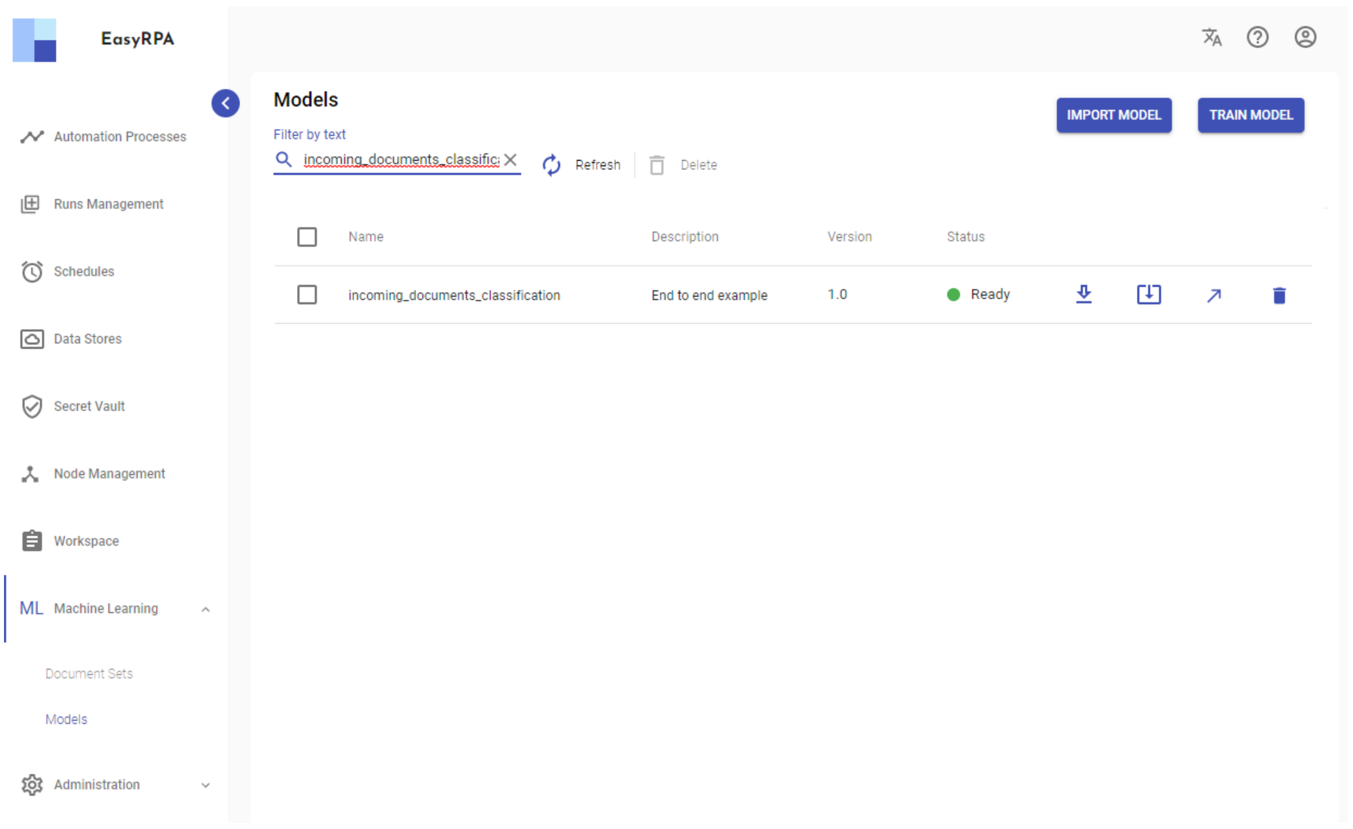
It has the following structure:

model.json

```
{
 "model_type": "textcat",
 "train_config": {
  "single_label": true,
  "iterations": 5
 }
}
```

- **model_type** (string) - for classification model it should be always "**textcat**" value
- **train_config** (object)
    - **single_label** (boolean) - specify if one document on out training set can have multiple classes (so multi-classes model will be trained)
    - **iterations** (integer) - with each iteration model updates weights. Default value is 30

After you click "TRAIN" button on confirmation dialog - wait until model training is finished and your model becomes with status "Ready". You can find the details on **Machine Learning Models** page:



Models

incoming_documents_classific

Name	Description	Version	Status
incoming_documents_classification	End to end example	1.0	● Ready

(v. 2.2) Develop Automation Process (Classification)

(v. 2.2) Step 1. Prepare input documents (Classification)

This step is completely the same as for Information Extraction: (v. 2.2) Step 1. Prepare input documents (IE)

Code example

In the same way, as in Information Extraction step, we're going to implement EasyRPA Storage scanning for new incoming invoices.

Also Automation Process looks the same except package names, AP name and configs:

InvoiceClassificationSample.java

```java
package eu.ibagroup.samples.classification.task;

import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModule;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import eu.ibagroup.samples.classification.task.GetIncomingInvoices;
import lombok.extern.slf4j.Slf4j;

@ApModuleEntry(name = "InvoiceClassificationSample")
@Slf4j
public class InvoiceClassificationSample extends ApModule {

    public TaskOutput run() throws Exception {
        TaskOutput incomingInvoicesResult = execute(getInput(), GetIncomingInvoices.class).get();
        return doNotCareOfResult();
    }
}
```

And Task by itself which scan all PDF files from specific folder in file storage and then move it to different working folder:

GetIncomingInvoices.java

```java
package eu.ibagroup.samples.classification.task;

import eu.ibagroup.easyrpa.engine.annotation.AfterInit;
import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.annotation.InputToOutput;
import eu.ibagroup.easyrpa.engine.annotation.Output;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.utils.storage.StorageManager;
import eu.ibagroup.samples.classification.ExportConstants;
import eu.ibagroup.samples.classification.PropertyConstants;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.io.FilenameUtils;

import javax.inject.Inject;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

@ApTaskEntry(name = "Get Documents from Storage")
@Slf4j
@InputToOutput
public class GetIncomingInvoices extends ApTask {

    @Inject
    private StorageManager storageManager;

    @Output(ExportConstants.INVOICE_DOCUMENT_S3_PATHS_LIST)
    private List<String> invoicesS3PathsList = new ArrayList<>();
    private String s3Bucket;
    private String s3FolderToScan;

    private String ocrS3WorkBucket;
    private String ocrS3WorkFolder;

    @AfterInit
    public void init() {
        s3Bucket = getConfigurationService().getConfiguration().get(PropertyConstants.S3_BUCKET);
        s3FolderToScan = getConfigurationService().getConfiguration().get(PropertyConstants.S3_FOLDER_TO_SCAN);

        ocrS3WorkBucket = getConfigurationService().getConfiguration().get(PropertyConstants.
OCR_S3_WORK_BUCKET);
        ocrS3WorkFolder = getConfigurationService().getConfiguration().get(PropertyConstants.
OCR_S3_WORK_FOLDER);
    }

    @Override
    public void execute() throws Exception {
        List<String> filesPathList = storageManager.listFiles(s3Bucket, s3FolderToScan, ".*\\.pdf");
        log.info("There're {} document(s) found to process on S3 bucket '{}' in folder '{}'", filesPathList.
size(), s3Bucket, s3FolderToScan);
        for (String s3Path : filesPathList) {
            String newS3Path = moveFile(s3Bucket, s3Path, ocrS3WorkBucket, ocrS3WorkFolder);
            invoicesS3PathsList.add(newS3Path);
        }
    }

    private String moveFile(String s3SourceBucket, String s3SourceFilePath, String s3DestBucket, String
s3DestFolder) throws IOException {
        String fileName = FilenameUtils.getName(s3SourceFilePath);
        String resultS3Path = s3DestFolder + "/" + fileName;
        copyFile(s3SourceBucket, s3SourceFilePath, s3DestBucket, resultS3Path);
        storageManager.deleteFile(s3SourceBucket, s3SourceFilePath);
        return resultS3Path;
    }

    private void copyFile(String s3SourceBucket, String s3SourceFilePath, String s3DestBucket, String
s3DestFilePath) throws IOException {
        try(InputStream fileInputStream = storageManager.getFile(s3SourceBucket, s3SourceFilePath)) {
            storageManager.uploadFile(s3DestBucket, s3DestFilePath, fileInputStream);
        }
    }
}
```

PropertyConstants.java

```
package eu.ibagroup.samples.classification;

public interface PropertyConstants {
    String S3_BUCKET = "s3.bucket";
    String S3_FOLDER_TO_SCAN = "s3.folder_to_scan";

    String OCR_S3_WORK_FOLDER = "ocr.s3_work_folder";
    String OCR_S3_WORK_BUCKET = "ocr.s3_work_bucket";
}
```

ExportConstants.java

```
package eu.ibagroup.samples.classification;

public interface ExportConstants {
    String INVOICE_DOCUMENT_S3_PATHS_LIST = "INVOICE_DOCUMENT_S3_PATHS_LIST";
}
```

apm_run.properties

```
# EasyRPA Client Configuration File
s3.bucket=classification-sample
s3.folder_to_scan=incoming_invoices

ocr.s3_work_bucket=data
ocr.s3_work_folder=incoming_invoices_work_folder
```

(v. 2.2) Step 2. Digitize documents using OCR (Classification)

This step is completely the same as for Information Extraction: (v. 2.2) Step 2. Digitize documents using OCR (IE)

Code example

Automation Process until this step looks the same as for Information Extraction except package names, AP name and configs:

AP class:

InvoiceClassificationSample.java

```
package eu.ibagroup.samples.classification;

import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModule;
import eu.ibagroup.easyrpa.engine.apflow.TaskInput;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrTask;
import eu.ibagroup.easyrpa.utils.CompletableFutures;
import eu.ibagroup.samples.classification.task.GetIncomingInvoices;
import eu.ibagroup.samples.classification.task.PrepareInputForOcrTask;
import lombok.extern.slf4j.Slf4j;

import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;

@ApModuleEntry(name = "InvoiceClassificationSample")
@Slf4j
public class InvoiceClassificationSample extends ApModule {

    public TaskOutput run() throws Exception {
        TaskOutput incomingInvoicesResult = execute(getInput(), GetIncomingInvoices.class).get();
        List<String> invoiceDocumentS3PathList = incomingInvoicesResult.get(ExportConstants.
INVOICE_DOCUMENT_S3_PATHS_LIST, List.class);

        List<CompletableFuture<TaskOutput>> invoiceProcessingExecutions = new ArrayList<>();
        if(invoiceDocumentS3PathList.size() > 0) {
            for(String documentS3Path : invoiceDocumentS3PathList) {
                TaskInput documentProcessingInput = new TaskInput(incomingInvoicesResult);
                documentProcessingInput.set(ExportConstants.INVOICE_DOCUMENT_S3_PATH, documentS3Path);

                CompletableFuture<TaskOutput> singleInvoiceProcessingExecution =
                        execute(documentProcessingInput, PrepareInputForOcrTask.class)
                                .thenCompose(execute(OcrTask.class));

                invoiceProcessingExecutions.add(singleInvoiceProcessingExecution);
            }
            CompletableFutures.allOf(invoiceProcessingExecutions).get();
        } else {
            log.info("No invoices for processing found");
        }

        return doNotCareOfResult();
    }
}
```

New class which prepare an input for OCR task:

PrepareInputForOcrTask.java

```
package eu.ibagroup.samples.classification.task;

import eu.ibagroup.easyrpa.engine.annotation.*;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrFormats;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrTask;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrTaskData;
import eu.ibagroup.easyrpa.utils.MlTaskUtils;
import eu.ibagroup.samples.classification.ExportConstants;
import eu.ibagroup.samples.classification.PropertyConstants;
import lombok.extern.slf4j.Slf4j;

import java.util.*;

@ApTaskEntry(name = "Prepare Input For OCR Task")
@Slf4j
@InputToOutput
public class PrepareInputForOcrTask extends ApTask {
    private static final String OCR_TASK_CONFIGURATION_WORK_S3_BUCKET_KEY = "documents_bucket";
    private static final String OCR_TASK_CONFIGURATION_TESSERACT_OPTIONS_KEY = "tesseractOptions";
    private static final String OCR_TASK_CONFIGURATION_IMAGE_MAGICK_OPTIONS_KEY = "imageMagickOptions";
    private static final List<String> OCR_OUTPUT_FORMATS = Arrays.asList(OcrFormats.TEXT, OcrFormats.HOCR,
OcrFormats.JSON, OcrFormats.IMAGE);

    @Input(ExportConstants.INVOICE_DOCUMENT_S3_PATH)
    private String documentS3Path;

    @Output(OcrTask.OCR_TASK_DATA_KEY)
    private OcrTaskData ocrTaskData;

    private String ocrS3WorkBucket;
    private String ocrTesseractOptions;
    private String ocrImageMagickOptions;

    private Map<String, Object> ocrConfiguration = new HashMap<>();

    @AfterInit
    public void init() {
        ocrS3WorkBucket = getConfigurationService().getConfiguration().get(PropertyConstants.
OCR_S3_WORK_BUCKET);
        ocrTesseractOptions = getConfigurationService().getConfiguration().get(PropertyConstants.
OCR_TESSERACT_OPTIONS);
        ocrImageMagickOptions = getConfigurationService().getConfiguration().get(PropertyConstants.
OCR_IMAGE_MAGICK_OPTIONS);

        ocrConfiguration.put(OCR_TASK_CONFIGURATION_WORK_S3_BUCKET_KEY, ocrS3WorkBucket);
        ocrConfiguration.put(OCR_TASK_CONFIGURATION_TESSERACT_OPTIONS_KEY, MlTaskUtils.readObjectFromString
(ocrTesseractOptions, ArrayList.class));
        ocrConfiguration.put(OCR_TASK_CONFIGURATION_IMAGE_MAGICK_OPTIONS_KEY, MlTaskUtils.readObjectFromString
(ocrImageMagickOptions, ArrayList.class));
    }

    @Override
    public void execute() throws Exception {
        log.info("Prepare input for document in '{}' with ocr task with configuration: {}", documentS3Path,
ocrConfiguration);

        ocrTaskData = new OcrTaskData();
        ocrTaskData.setDocumentLocation(documentS3Path);
        ocrTaskData.getFormats().addAll(OCR_OUTPUT_FORMATS);
        ocrTaskData.setConfiguration(ocrConfiguration);
    }
}
```

PropertyConstants.java

```
package eu.ibagroup.samples.classification;

public interface PropertyConstants {
    String S3_BUCKET = "s3.bucket";
    String S3_FOLDER_TO_SCAN = "s3.folder_to_scan";

    String OCR_S3_WORK_FOLDER = "ocr.s3_work_folder";
    String OCR_S3_WORK_BUCKET = "ocr.s3_work_bucket";
    String OCR_TESSERACT_OPTIONS = "ocr.tesseract_options";
    String OCR_IMAGE_MAGICK_OPTIONS = "ocr.image_magick_options";
}
```

ExportConstants.java

```
package eu.ibagroup.samples.classification;

public interface ExportConstants {
    String INVOICE_DOCUMENT_S3_PATHS_LIST = "INVOICE_DOCUMENT_S3_PATHS_LIST";
    String INVOICE_DOCUMENT_S3_PATH = "INVOICE_DOCUMENT_S3_PATH";
}
```

apm_run.properties

```
# EasyRPA Client Configuration File
s3.bucket=classification-sample
s3.folder_to_scan=incoming_invoices

ocr.s3_work_bucket=data
ocr.s3_work_folder=incoming_invoices_work_folder
ocr.tesseract_options=["-l", "eng","--psm","3","--oem","3","--dpi", "800"]
ocr.image_magick_options=["-resample", "450", "-density", "350", "-quality", "100", "-background" , "white" , "-
alpha" , "flatten"]
```

(v. 2.2) Step 3. Apply and execute ML model (Classification)

This step is completely the same as for Information Extraction: (v. 2.2) Step 3. Apply and execute ML model (IE)

Code example

Automation Process until this step looks the same as for Information Extraction except package names, AP name and configs:

InvoiceProcessingSample.java

```
package eu.ibagroup.samples.classification;

import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModule;
import eu.ibagroup.easyrpa.engine.apflow.TaskInput;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import eu.ibagroup.easyrpa.engine.task.ml.MlTask;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrTask;
import eu.ibagroup.easyrpa.utils.CompletableFutures;
import eu.ibagroup.samples.classification.task.GetIncomingInvoices;
import eu.ibagroup.samples.classification.task.PrepareInputForMlTask;
import eu.ibagroup.samples.classification.task.PrepareInputForOcrTask;
import lombok.extern.slf4j.Slf4j;

import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;

@ApModuleEntry(name = "InvoiceClassificationSample")
@Slf4j
public class InvoiceClassificationSample extends ApModule {

    public TaskOutput run() throws Exception {
        TaskOutput incomingInvoicesResult = execute(getInput(), GetIncomingInvoices.class).get();
        List<String> invoiceDocumentS3PathList = incomingInvoicesResult.get(ExportConstants.
INVOICE_DOCUMENT_S3_PATHS_LIST, List.class);

        List<CompletableFuture<TaskOutput>> invoiceProcessingExecutions = new ArrayList<>();
        if(invoiceDocumentS3PathList.size() > 0) {
            for(String documentS3Path : invoiceDocumentS3PathList) {
                TaskInput documentProcessingInput = new TaskInput(incomingInvoicesResult);
                documentProcessingInput.set(ExportConstants.INVOICE_DOCUMENT_S3_PATH, documentS3Path);

                CompletableFuture<TaskOutput> singleInvoiceProcessingExecution =
                        execute(documentProcessingInput, PrepareInputForOcrTask.class)
                                .thenCompose(execute(OcrTask.class))
                                .thenCompose(execute(PrepareInputForMlTask.class))
                                .thenCompose(execute(MlTask.class));

                invoiceProcessingExecutions.add(singleInvoiceProcessingExecution);
            }
            CompletableFutures.allOf(invoiceProcessingExecutions).get();
        } else {
            log.info("No invoices for processing found");
        }

        return doNotCareOfResult();
    }

}
```

Code of **PrepareInputForMlTask:**

PrepareInputForMlTask.java

```java
package eu.ibagroup.samples.classification.task;

import eu.ibagroup.easyrpa.engine.annotation.*;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.engine.task.ml.MlTask;
import eu.ibagroup.easyrpa.engine.task.ml.MlTaskData;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrTask;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrTaskData;
import eu.ibagroup.easyrpa.utils.MlTaskUtils;
import eu.ibagroup.samples.classification.ExportConstants;
import eu.ibagroup.samples.classification.PropertyConstants;
import lombok.extern.slf4j.Slf4j;

import java.util.HashMap;
import java.util.Map;
import java.util.UUID;

@ApTaskEntry(name = "Prepare input for ML Task")
@Slf4j
@InputToOutput
public class PrepareInputForMlTask extends ApTask {
    private static final String ML_TASK_CONFIGURATION_WORK_S3_BUCKET_KEY = "documents_bucket";

    @Input(OcrTask.OCR_TASK_DATA_KEY)
    private OcrTaskData ocrTaskOutput;

    @Input(ExportConstants.INVOICE_DOCUMENT_S3_PATH)
    private String documentS3Path;

    @Output(MlTask.ML_TASK_DATA_KEY)
    private MlTaskData mlTaskData;

    private String modelName;
    private String modelVersion;

    private String documentId;

    private Map<String, Object> mlConfiguration = new HashMap<>();

    @AfterInit
    public void init() {
        modelName = getConfigurationService().get(PropertyConstants.CLASSIFICATION_MODEL_NAME);
        modelVersion = getConfigurationService().get(PropertyConstants.CLASSIFICATION_MODEL_VERSION);

        documentId = UUID.randomUUID().toString();

        String ocrS3WorkBucket = getConfigurationService().getConfiguration().get(PropertyConstants.
OCR_S3_WORK_BUCKET);
        mlConfiguration.put(ML_TASK_CONFIGURATION_WORK_S3_BUCKET_KEY, ocrS3WorkBucket);
    }

    @Override
    public void execute() {
        log.info("Creating ML task input for document located at '{}' with assigned id '{}'", documentS3Path,
documentId);

        mlTaskData = new MlTaskData();
        mlTaskData.setModelName(modelName);
        mlTaskData.setModelVersion(modelVersion);
        mlTaskData.setConfiguration(mlConfiguration);
        MlTaskUtils.prepareClMlTask(documentId, mlTaskData, ocrTaskOutput);
    }

}
```

PropertyConstants.java

```
package eu.ibagroup.samples.classification;

public interface PropertyConstants {
    String S3_BUCKET = "s3.bucket";
    String S3_FOLDER_TO_SCAN = "s3.folder_to_scan";

    String OCR_S3_WORK_FOLDER = "ocr.s3_work_folder";
    String OCR_S3_WORK_BUCKET = "ocr.s3_work_bucket";
    String OCR_TESSERACT_OPTIONS = "ocr.tesseract_options";
    String OCR_IMAGE_MAGICK_OPTIONS = "ocr.image_magick_options";

          String CLASSIFICATION_MODEL_NAME = "classification_model.name";
        String CLASSIFICATION_MODEL_VERSION = "classification_model.version";
}
```

ExportConstants.java

```
package eu.ibagroup.samples.classification;

public interface ExportConstants {
    String INVOICE_DOCUMENT_S3_PATHS_LIST = "INVOICE_DOCUMENT_S3_PATHS_LIST";
    String INVOICE_DOCUMENT_S3_PATH = "INVOICE_DOCUMENT_S3_PATH";
}
```

apm_run.properties

```
# EasyRPA Client Configuration File
s3.bucket=ie-sample
s3.folder_to_scan=incoming_invoices

ocr.s3_work_bucket=data
ocr.s3_work_folder=incoming_invoices_work_folder
ocr.tesseract_options=["-l", "eng","--psm","3","--oem","3","--dpi", "800"]
ocr.image_magick_options=["-resample", "450", "-density", "350", "-quality", "100", "-background" , "white" , "-
alpha" , "flatten"]

classification_model.name=incoming_documents_classification
classification_model.version=1.0
```

(v. 2.2) Step 4. Read ML output and route documents to Human Task (Classification)

To read the ML output data you should get value by "**ml.task.data**" key from **TaskOutput** object. It contains **MlTaskData** object, which contains result data, so you can the final category and its score.

There is special method **MlTaskUtils.createClHtOutputMap(mlTaskData)** which transform classification ML output to Map:

```
MlTaskData mlTaskDataOutput = previousTaskStepOutput.get(MlTask.ML_TASK_DATA_KEY, MlTaskData.class);
Map classificationResult = MlTaskUtils.createClHtOutputMap(mlTaskDataOutput);

String resultCategory = ((List)classificationResult.get("categories")).get(0).toString();
Double resultScore = (Double) ((Map)classificationResult.get("scores")).get(resultCategory);
```

Also usually if result category was classified with not enough confidence, we decide to **send document to Human Task**. You can read more about how to send document to Human task.

Basically to send document to Human Task you need to prepare special input for embedded **HumanTask** class. Input should contains **HumanTaskData** object under "**human.task.data**" key in input.

Code example

In the AP class we add additional function which is asynchronously executed and it reads output from ML step. If result category score is lower than confidence threshold (in our case 30%), we asynchronously execute Human Task. If category classified with enough confidence - we do nothing (we return already completed Completable Future with output from previous step):

InvoiceClassificationSample.java

```
package eu.ibagroup.samples.classification;

import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModule;
import eu.ibagroup.easyrpa.engine.apflow.TaskInput;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import eu.ibagroup.easyrpa.engine.task.ht.HumanTask;
import eu.ibagroup.easyrpa.engine.task.ml.MlTask;
import eu.ibagroup.easyrpa.engine.task.ml.MlTaskData;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrTask;
import eu.ibagroup.easyrpa.utils.CompletableFutures;
import eu.ibagroup.samples.classification.task.*;
import lombok.extern.slf4j.Slf4j;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.concurrent.CompletableFuture;

@ApModuleEntry(name = "InvoiceClassificationSample")
@Slf4j
public class InvoiceClassificationSample extends ApModule {
        private static final Double CLASSIFICATION_CONFIDENCE_THRESHOLD = 0.3;

    public TaskOutput run() throws Exception {
        TaskOutput incomingInvoicesResult = execute(getInput(), GetIncomingInvoices.class).get();
        List<String> invoiceDocumentS3PathList = incomingInvoicesResult.get(ExportConstants.
INVOICE_DOCUMENT_S3_PATHS_LIST, List.class);

        List<CompletableFuture<TaskOutput>> invoiceProcessingExecutions = new ArrayList<>();
        if(invoiceDocumentS3PathList.size() > 0) {
            for(String documentS3Path : invoiceDocumentS3PathList) {
                TaskInput documentProcessingInput = new TaskInput(incomingInvoicesResult);
                documentProcessingInput.set(ExportConstants.INVOICE_DOCUMENT_S3_PATH, documentS3Path);
                CompletableFuture<TaskOutput> singleInvoiceProcessingExecution =
                        execute(documentProcessingInput, PrepareInputForOcrTask.class)
                                .thenCompose(execute(OcrTask.class))
                                .thenCompose(execute(PrepareInputForMlTask.class))
                                .thenCompose(execute(MlTask.class))
                                .thenCompose(previousTaskStepOutput -> {
                                    MlTaskData mlTaskDataOutput = previousTaskStepOutput.get(MlTask.
ML_TASK_DATA_KEY, MlTaskData.class);
                                                                        Map classificationResult = MlTaskUtils.
createClHtOutputMap(mlTaskDataOutput);

                                                                        String resultCategory = ((List)
classificationResult.get("categories")).get(0).toString();
                                                                        Double resultScore = (Double) ((Map)
classificationResult.get("scores")).get(resultCategory);

                                                                        if(resultScore <
CLASSIFICATION_CONFIDENCE_THRESHOLD) {
                                                                                return execute
(previousTaskStepOutput, PrepareInputForHumanTask.class)
                                                                                                .thenCompose
(execute(HumanTask.class));
                                                                        } else {
                                                                                return CompletableFuture.
completedFuture(previousTaskStepOutput);
                                                                        }
                                });

                invoiceProcessingExecutions.add(singleInvoiceProcessingExecution);
            }
            CompletableFutures.allOf(invoiceProcessingExecutions).get();
        } else {
            log.info("No invoices for processing found");
        }

        return doNotCareOfResult();
    }
}
```

To send document to Human Task, we add additional step which is responsible for preparing input data for it. Additionally we special boolean variable to output which helps on next steps to identify if record was processed by Human or not.

PrepareInputForHumanTask.java

```
package eu.ibagroup.samples.classification.task;

import eu.ibagroup.easyrpa.engine.annotation.*;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.engine.task.ht.HumanTask;
import eu.ibagroup.easyrpa.engine.task.ht.HumanTaskData;
import eu.ibagroup.easyrpa.engine.task.ml.MlTask;
import eu.ibagroup.easyrpa.engine.task.ml.MlTaskData;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrTask;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrTaskData;
import eu.ibagroup.easyrpa.utils.MlTaskUtils;
import eu.ibagroup.easyrpa.utils.storage.StorageManager;
import eu.ibagroup.samples.classification.ExportConstants;
import eu.ibagroup.samples.classification.PropertyConstants;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.io.FilenameUtils;

import javax.inject.Inject;
import java.util.Map;

@ApTaskEntry(name = "Prepare Human Task")
@Slf4j
@InputToOutput
public class PrepareInputForHumanTask extends ApTask  {
    @Inject
    private StorageManager storageManager;

    @Input(OcrTask.OCR_TASK_DATA_KEY)
    private OcrTaskData ocrTaskOutput;

    @Input(MlTask.ML_TASK_DATA_KEY)
    private MlTaskData mlTaskOutput;

    @Input(ExportConstants.INVOICE_DOCUMENT_S3_PATH)
    private String documentS3Path;

    @Output(HumanTask.HUMAN_TASK_DATA_KEY)
    private HumanTaskData humanTaskData;

    @Output(ExportConstants.DOCUMENT_PROCESSED_BY_HUMAN)
    private boolean documentProcessedByHuman = true;

    private String ocrS3WorkBucket;
    private String documentType;

    @AfterInit
    public void init() {
        ocrS3WorkBucket = getConfigurationService().getConfiguration().get(PropertyConstants.
OCR_S3_WORK_BUCKET);
        documentType = getConfigurationService().getConfiguration().get(PropertyConstants.DOCUMENT_TYPE);
    }

    @Override
    public void execute() throws Exception {
        String documentName = FilenameUtils.getName(documentS3Path);
        log.info("Creating human task for document {} with document type {}", documentName, documentType);

        humanTaskData = new HumanTaskData();
                 humanTaskData.setInputJson(MlTaskUtils.createClHtInputMap(ocrTaskOutput, storageManager,
ocrS3WorkBucket));
                humanTaskData.setOutputJson(MlTaskUtils.createClHtOutputMap(mlTaskOutput));

        humanTaskData.setName(documentName);
        humanTaskData.setDocumentType(documentType);
        humanTaskData.setDescription("Invoice Document: " + documentName);

        log.info("Sending task into workspace {} ", humanTaskData);
    }

}
```

PropertyConstants.java

```
package eu.ibagroup.samples.ie;

public interface PropertyConstants {
    String S3_BUCKET = "s3.bucket";
    String S3_FOLDER_TO_SCAN = "s3.folder_to_scan";

    String OCR_S3_WORK_FOLDER = "ocr.s3_work_folder";
    String OCR_S3_WORK_BUCKET = "ocr.s3_work_bucket";
    String OCR_TESSERACT_OPTIONS = "ocr.tesseract_options";
    String OCR_IMAGE_MAGICK_OPTIONS = "ocr.image_magick_options";

    String IE_MODEL_NAME = "ie_model.name";
    String IE_MODEL_VERSION = "ie_model.version";

    String DOCUMENT_TYPE = "document_type";
}
```

ExportConstants.java

```
package eu.ibagroup.samples.classification;

public interface ExportConstants {
    String INVOICE_DOCUMENT_S3_PATHS_LIST = "INVOICE_DOCUMENT_S3_PATHS_LIST";
    String INVOICE_DOCUMENT_S3_PATH = "INVOICE_DOCUMENT_S3_PATH";
    String DOCUMENT_PROCESSED_BY_HUMAN = "DOCUMENT_PROCESSED_BY_HUMAN";
}
```

apm_run.properties

```
# EasyRPA Client Configuration File
s3.bucket=classification-sample
s3.folder_to_scan=incoming_invoices

ocr.s3_work_bucket=data
ocr.s3_work_folder=incoming_invoices_work_folder
ocr.tesseract_options=["-l", "eng","--psm","3","--oem","3","--dpi", "800"]
ocr.image_magick_options=["-resample", "450", "-density", "350", "-quality", "100", "-background" , "white" , "-
alpha" , "flatten"]

classification_model.name=incoming_documents_classification
classification_model.version=1.0

document_type=Incoming Documents Classification
```

(v. 2.2) Step 5. Process extraction result (Classification)

After document is classified by ML or with help from Human, usually in business processes we route the documents to different flow depends on category.

The main goal of this step is to demonstrate how we get extracted category depends on classification result source (ML or Human Task).
As we have a special boolean output variable which we added on previous step, we can easily determine if document processed by Human or ML, so depends on that there're different ways to extract result:

```
if(documentProcessedByHuman) {
        resultCategory = ((List)humanTaskOutput.getOutputJson().get("categories")).get(0).toString();
        resultScore = 1d; //confidence is 100% as it was processed by human
} else {
        Map classificationResult = MlTaskUtils.createClHtOutputMap(mlTaskOutput);

        resultCategory = ((List)classificationResult.get("categories")).get(0).toString();
        resultScore = (Double) ((Map)classificationResult.get("scores")).get(resultCategory);
}
```

Code example

The final AP code is:

InvoiceClassificationSample.java

```
package eu.ibagroup.samples.classification;

import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModule;
import eu.ibagroup.easyrpa.engine.apflow.TaskInput;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import eu.ibagroup.easyrpa.engine.task.ht.HumanTask;
import eu.ibagroup.easyrpa.engine.task.ml.MlTask;
import eu.ibagroup.easyrpa.engine.task.ml.MlTaskData;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrTask;
import eu.ibagroup.easyrpa.utils.CompletableFutures;
import eu.ibagroup.easyrpa.utils.MlTaskUtils;
import eu.ibagroup.samples.classification.task.*;
import lombok.extern.slf4j.Slf4j;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.concurrent.CompletableFuture;
import java.util.function.Function;

@ApModuleEntry(name = "InvoiceClassificationSample")
@Slf4j
public class InvoiceClassificationSample extends ApModule {
    private static final Double CLASSIFICATION_CONFIDENCE_THRESHOLD = 0.3;

    public TaskOutput run() throws Exception {
        TaskOutput incomingInvoicesResult = execute(getInput(), GetIncomingInvoices.class).get();
        List<String> invoiceDocumentS3PathList = incomingInvoicesResult.get(ExportConstants.
INVOICE_DOCUMENT_S3_PATHS_LIST, List.class);

        List<CompletableFuture<TaskOutput>> invoiceProcessingExecutions = new ArrayList<>();
        if(invoiceDocumentS3PathList.size() > 0) {
            for(String documentS3Path : invoiceDocumentS3PathList) {
                TaskInput documentProcessingInput = new TaskInput(incomingInvoicesResult);
                documentProcessingInput.set(ExportConstants.INVOICE_DOCUMENT_S3_PATH, documentS3Path);

                CompletableFuture<TaskOutput> singleInvoiceProcessingExecution =
                        execute(documentProcessingInput, PrepareInputForOcrTask.class)
```

```
                                .thenCompose(execute(OcrTask.class))
                                .thenCompose(execute(PrepareInputForMlTask.class))
                                .thenCompose(execute(MlTask.class))
                                .thenCompose(processAndRouteMlTaskResult())
                                .thenCompose(execute(PrepareResult.class));

                invoiceProcessingExecutions.add(singleInvoiceProcessingExecution);
            }
            CompletableFutures.allOf(invoiceProcessingExecutions).get();
        } else {
            log.info("No invoices for processing found");
        }

        return doNotCareOfResult();
    }

    private Function<TaskOutput, CompletableFuture<TaskOutput>> processAndRouteMlTaskResult() {
        return previousTaskStepOutput -> {
            MlTaskData mlTaskDataOutput = previousTaskStepOutput.get(MlTask.ML_TASK_DATA_KEY, MlTaskData.class);
            Map classificationResult = MlTaskUtils.createClHtOutputMap(mlTaskDataOutput);

            String resultCategory = ((List)classificationResult.get("categories")).get(0).toString();
            Double resultScore = (Double) ((Map)classificationResult.get("scores")).get(resultCategory);

            if(resultScore < CLASSIFICATION_CONFIDENCE_THRESHOLD) {
                return execute(previousTaskStepOutput, PrepareInputForHumanTask.class)
                        .thenCompose(execute(HumanTask.class));
            } else {
                return CompletableFuture.completedFuture(previousTaskStepOutput);
            }
        };
    }
}
```

Final PrepareResult step, it generates an output with final text:

PrepareResult.java

```
package eu.ibagroup.samples.classification.task;

import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.annotation.Input;
import eu.ibagroup.easyrpa.engine.annotation.InputToOutput;
import eu.ibagroup.easyrpa.engine.annotation.Output;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.engine.task.ht.HumanTask;
import eu.ibagroup.easyrpa.engine.task.ht.HumanTaskData;
import eu.ibagroup.easyrpa.engine.task.ml.MlTask;
import eu.ibagroup.easyrpa.engine.task.ml.MlTaskData;
import eu.ibagroup.easyrpa.utils.MlTaskUtils;
import eu.ibagroup.samples.classification.ExportConstants;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.io.FilenameUtils;

import java.util.List;
import java.util.Map;

@ApTaskEntry(name = "Validate invoice data")
@Slf4j
@InputToOutput
public class PrepareResult extends ApTask {

    @Input(value = ExportConstants.DOCUMENT_PROCESSED_BY_HUMAN, required = false)
    private boolean documentProcessedByHuman;

    @Input(value = HumanTask.HUMAN_TASK_DATA_KEY, required = false)
    private HumanTaskData humanTaskOutput;

    @Input(value = MlTask.ML_TASK_DATA_KEY, required = false)
    private MlTaskData mlTaskOutput;

    @Input(ExportConstants.INVOICE_DOCUMENT_S3_PATH)
    private String documentS3Path;

    @Output(ExportConstants.PROCESSING_RESULT)
    private String processingResult;

    @Override
    public void execute() throws Exception {
        String documentName = FilenameUtils.getName(documentS3Path);
        String resultCategory;
        Double resultScore;
        if(documentProcessedByHuman) {
            resultCategory = ((List)humanTaskOutput.getOutputJson().get("categories")).get(0).toString();
            resultScore = 1d; //confidence is 100% as it was processed by human
        } else {
            Map classificationResult = MlTaskUtils.createClHtOutputMap(mlTaskOutput);

            resultCategory = ((List)classificationResult.get("categories")).get(0).toString();
            resultScore = (Double) ((Map)classificationResult.get("scores")).get(resultCategory);
        }

        processingResult = "Classification result: document '" + documentName + "' relates to category '" +
resultCategory + "' with confidence " + resultScore;
        log.info(processingResult);
    }
}
```

PropertyConstants.java

```
package eu.ibagroup.samples.classification;

public interface PropertyConstants {
        String S3_BUCKET = "s3.bucket";
        String S3_FOLDER_TO_SCAN = "s3.folder_to_scan";

        String OCR_S3_WORK_FOLDER = "ocr.s3_work_folder";
        String OCR_S3_WORK_BUCKET = "ocr.s3_work_bucket";
        String OCR_TESSERACT_OPTIONS = "ocr.tesseract_options";
        String OCR_IMAGE_MAGICK_OPTIONS = "ocr.image_magick_options";

        String CLASSIFICATION_MODEL_NAME = "classification_model.name";
        String CLASSIFICATION_MODEL_VERSION = "classification_model.version";

        String DOCUMENT_TYPE = "document_type";
}
```

ExportConstants.java

```
package eu.ibagroup.samples.classification;

public interface ExportConstants {
        String INVOICE_DOCUMENT_S3_PATHS_LIST = "INVOICE_DOCUMENT_S3_PATHS_LIST";
        String INVOICE_DOCUMENT_S3_PATH = "INVOICE_DOCUMENT_S3_PATH";
        String DOCUMENT_PROCESSED_BY_HUMAN = "DOCUMENT_PROCESSED_BY_HUMAN";
        String PROCESSING_RESULT = "PROCESSING_RESULT";
}
```

apm_run.properties

```
# EasyRPA Client Configuration File
s3.bucket=classification-sample
s3.folder_to_scan=incoming_invoices
```

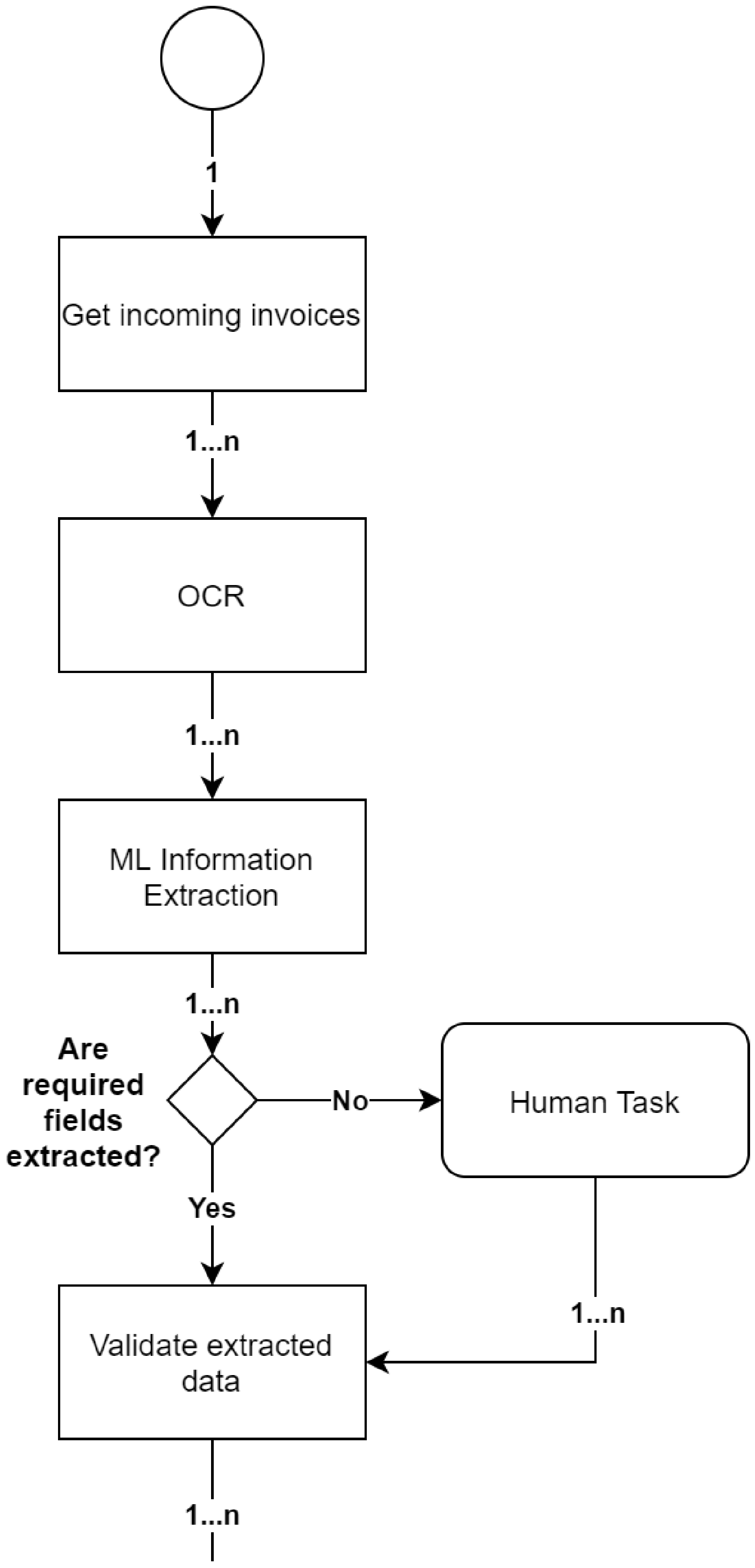
```
ocr.s3_work_bucket=data
ocr.s3_work_folder=incoming_invoices_work_folder
ocr.tesseract_options=["-l", "eng","--psm","3","--oem","3","--dpi", "800"]
ocr.image_magick_options=["-resample", "450", "-density", "350", "-quality", "100", "-background" , "white" , "-
alpha" , "flatten"]

classification_model.name=incoming_documents_classification
classification_model.version=1.0

document_type=Incoming Documents Classification
```

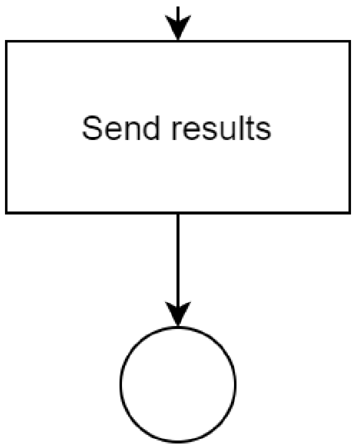
(v. 2.2) Information Extraction process

This article covers the end-to-end process of developing Information Extraction (IE) automation and contains 2 sections:

1. Setup Human Task and Train model **in 3 steps**
2. (v. 2.2) Develop Automation Process (IE) which use ML model **in 5 steps**

From the high-level perspective, the IE automation process looks like on the diagram below and it should process all records asynchronously in parallel:


1
Get incoming invoices
1...n
OCR
1...n
ML Information Extraction
1...n
Are required fields extracted?
No
Human Task
Yes
1...n
Validate extracted data
1...n



Analyze the document type to decide the next steps

Depends on the file format of your incoming documents, there're different ways we suggest working with it.

In the table below you may see 2 different suggested approaches:

- **Machine Learning** - if you decided to go with this approach, you can continue following this guide.
- **Rule-based** - if you decided to go with a rule-based approach - you can skip the next steps of this guide and implement the suggested way to extract valuable information from documents.

Document file format	Suggest approach	Comment
PDF	*Machine Learning*	PDF documents can be categorized in three different types, depending on the way the file originated. How it was originally created also defines whether the content of the PDF (text, images, tables) can be accessed or whether it is “locked” in an image of the page. ***1.“True” or Digitally Created PDFs*** Digitally created PDFs, also known as “true“ PDFs, are created using software such as Microsoft® Word, Excel® or via the “print” function within a software application (virtual printer). They consist of text and images. Both the characters in the text and the meta-information have an electronic character designation. With ABBYY FineReader 14 you can easily search through these PDFs and select, edit or delete text similar to how you would do that in other editable formats, such as Microsoft® Word. The images in digitally created documents can be resized, moved, or deleted. ***2. “Image-only” or Scanned PDFs*** When scanning hard copy documents on MFPs and office scanners, or when converting a camera image, jpg, tiff or screenshot into a PDF, the content is “locked” in a snapshot-like image. Such image-only PDF documents contain just the scanned/photographed images of pages, without an underlying text layer. Consequently, image-only PDF files are not searchable, and their text usually cannot be modified or marked up. An “image-only” PDF can be made searchable by applying OCR with which a text layer is added, normally under the page image. ***3. Searchable PDFs*** Searchable PDFs usually result through the application of OCR (Optical Character Recognition) to scanned PDFs or other image-based documents. During the text recognition process, characters and the document structure are analyzed and “read”. A text layer is added to the image layer, usually placed underneath. Such PDF files are almost indistinguishable from the original documents and are fully searchable. Text in searchable PDF documents can be selected, copied, and marked up.

		Processing pdf workflow depends on type of PDF. In case of Searchable (**case 3**) or "True" PDFs (**case 1**) we can get content of the files using **pdfbox** apache library. "Image-only" PDFs (**case 2**) should go through OCR step at first.
Image	*Machine Learning*	This type of documents usually comes as scans. Typical workflow for images is the following: 1. Make sure that an image has sufficient resolution (at least 300 dpi) and convert it if needed (using **I mageMagic** library) 2. OCR step
Excel	*Rule-based*	Both approaches (ML and Rule-based) can be applied for this type of document, but generally it is much easy to implement **rules-based approach** cause excel is structured document. ML approach can be used if the customer has many different templates of excel documents. Keep in mind that you should **convert excel to html** before sending the document to Manual Task
HTML	*Rule-based*	For html files we also can go with both approaches (**ML and Rule-based**). Please be aware that if customer has well structured HTML format probably the best solution is to **use xpath to extract data from documents** Only in case of you don't know the structure of HTML, you should use **ML approach**. Usually it happens when you need to extract the data from not auto generated email body by itself, as people write email without any predefined template.
Plain Text	*Machine Learning*	ML approach is preferred for this format, but rule-based also can be applied (for example we 100% sure that invoice number is the first word in the document). Please note that plain text is the worth case for the ML approach cause this format does not have any additional information (like html tags).
Other formats		You may encounter other types of documents. Review the structure of the documents to make the right decision on the use of the approach

(v. 2.2) Setup Human Task and Train ML model (IE)

(v. 2.2) Step 1. Create new Document Type (IE)

To describe which fields you want to extract from document you should create new **Document Type**.

Document Type is a configuration which lets Human Task and ML model know the output fields you want to receive from the document.

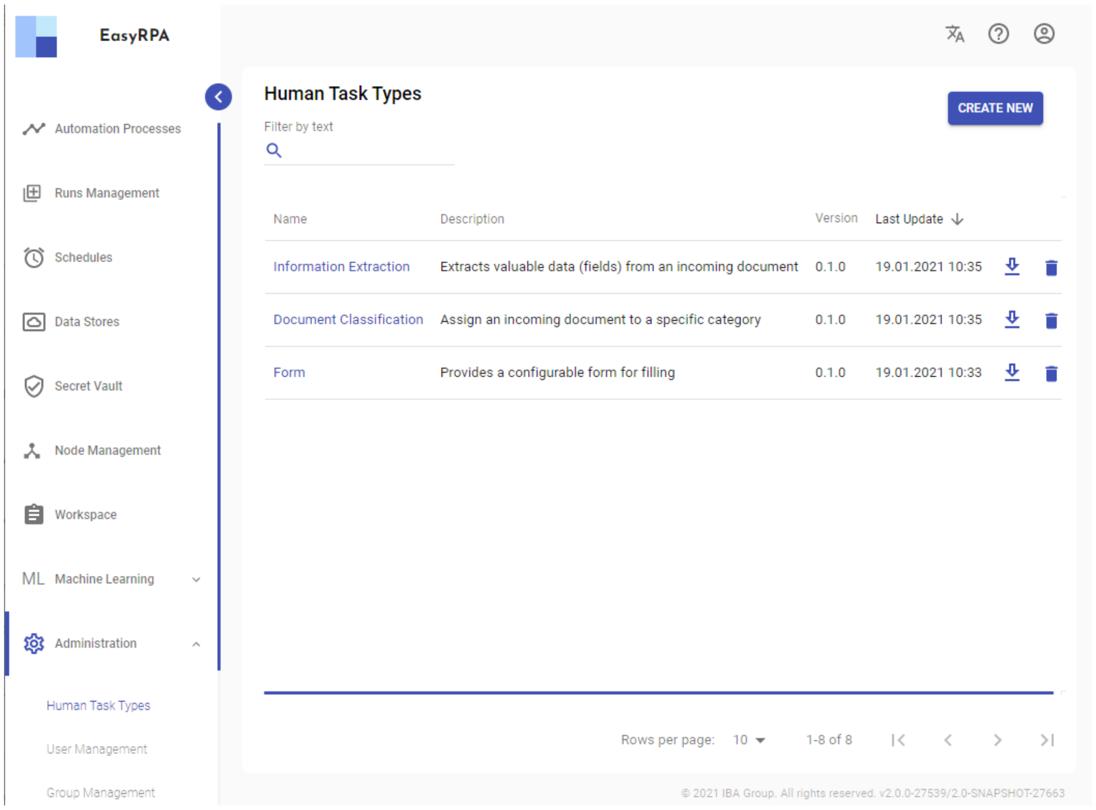
Each Document Type relates to some of the **Human Task Type**.

Human Task Type is a special small Web-application which can read and parse your Document Type configuration and it knows how to display input documents.

EasyRPA provides 3 predefined Human Task Types:

- Information Extraction
- Document Classification
- Form

For the Information Extraction process we need to create Document Type which relates to the **Information Extraction** predefined Human Task Type:



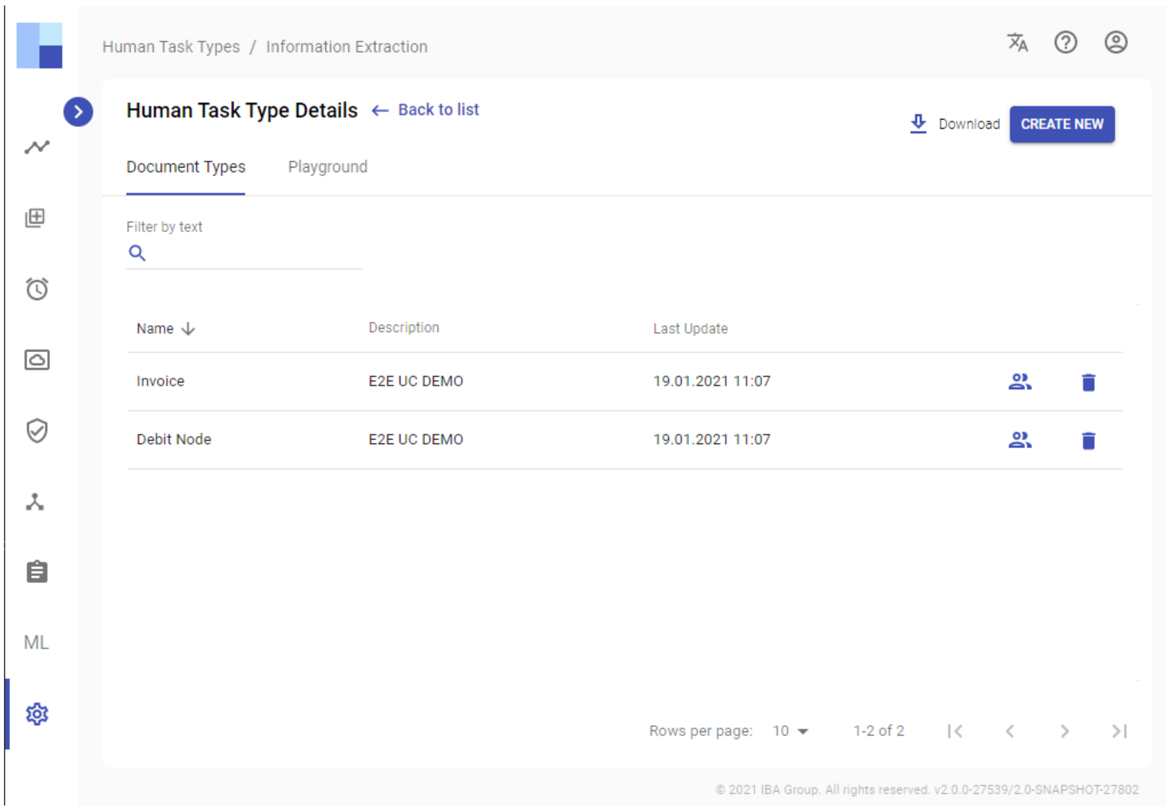
Human Task Types

Name	Description	Version	Last Update
Information Extraction	Extracts valuable data (fields) from an incoming document	0.1.0	19.01.2021 10:35
Document Classification	Assign an incoming document to a specific category	0.1.0	19.01.2021 10:35
Form	Provides a configurable form for filling	0.1.0	19.01.2021 10:33

© 2021 IBA Group. All rights reserved. v2.0.0-27539/2.0-SNAPSHOT-27663

** You're able to create your own Human Task Types for specific use-cases. You can read about at (v. 2.2) Create Human Task Type*

When you open Information Extraction type there's a list of all document types are related to that Human Task Type, and you will also see "CREATE NEW" button:


Human Task Types / Information Extraction

Human Task Type Details ← Back to list

Download CREATE NEW

Document Types | Playground

Filter by text

Name	Description	Last Update
Invoice	E2E UC DEMO	19.01.2021 11:07
Debit Node	E2E UC DEMO	19.01.2021 11:07

Rows per page: 10 1-2 of 2

© 2021 IBA Group. All rights reserved. v2.0.0-27539/2.0-SNAPSHOT-27802

You need to provide the name, description and settings for new Document Type.

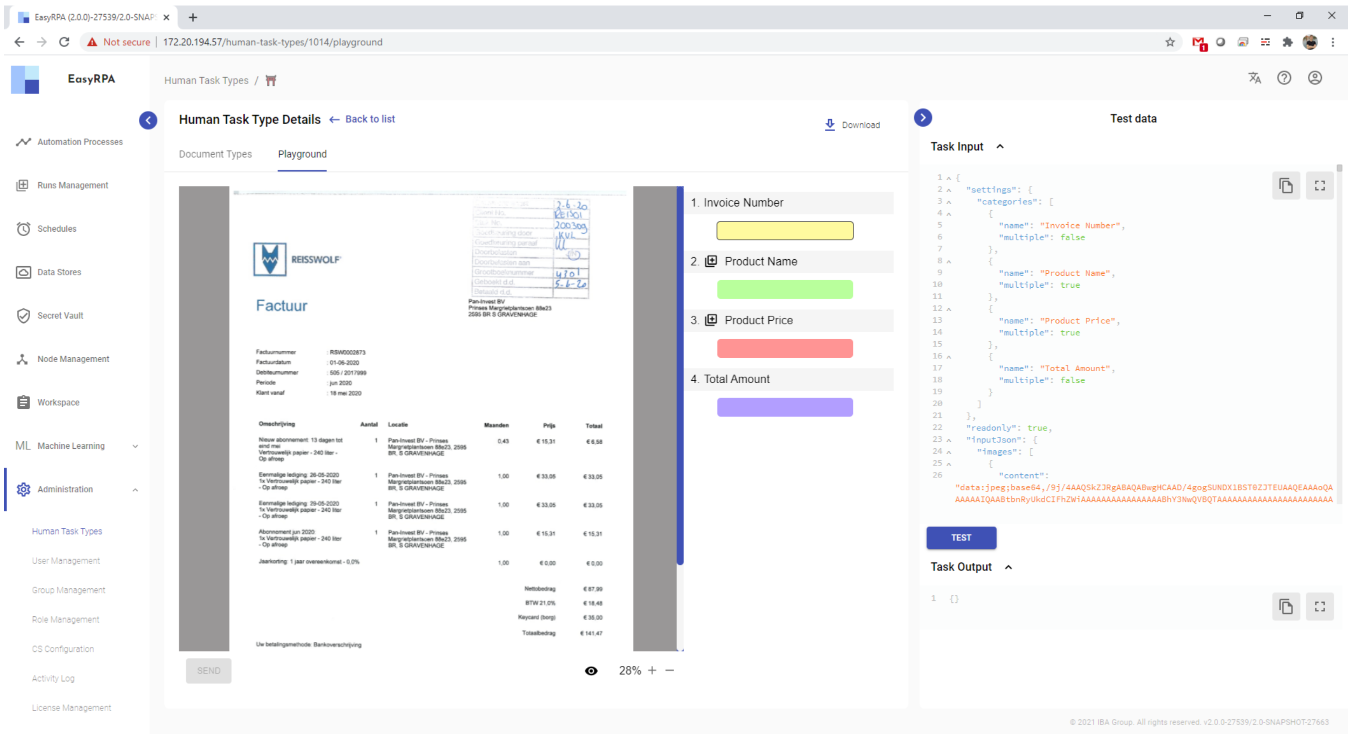
Settings is a configuration in JSON format which has the structure as in example bellow:

```
{
        "categories": [
                {
                        "name": "Invoice Number",
                        "multiple": false
                },
                {
                        "name": "Product Name",
                        "multiple": true
                },
                {
                        "name": "Product Price",
                        "multiple": true
                },
                {
                        "name": "Total Amount",
                        "multiple": false
                }
        ]
}
```

This setting contains list of "categories" items. Where each item contains:

- **name** (string) - it serves as a key to receive data from Human Task output
- **multiple** (boolean) - shows if this field may have multiple values (uses when you have the list with details of some items in your document, e.g. list of products)

The Document Type which is configured above will looks like on the following image:



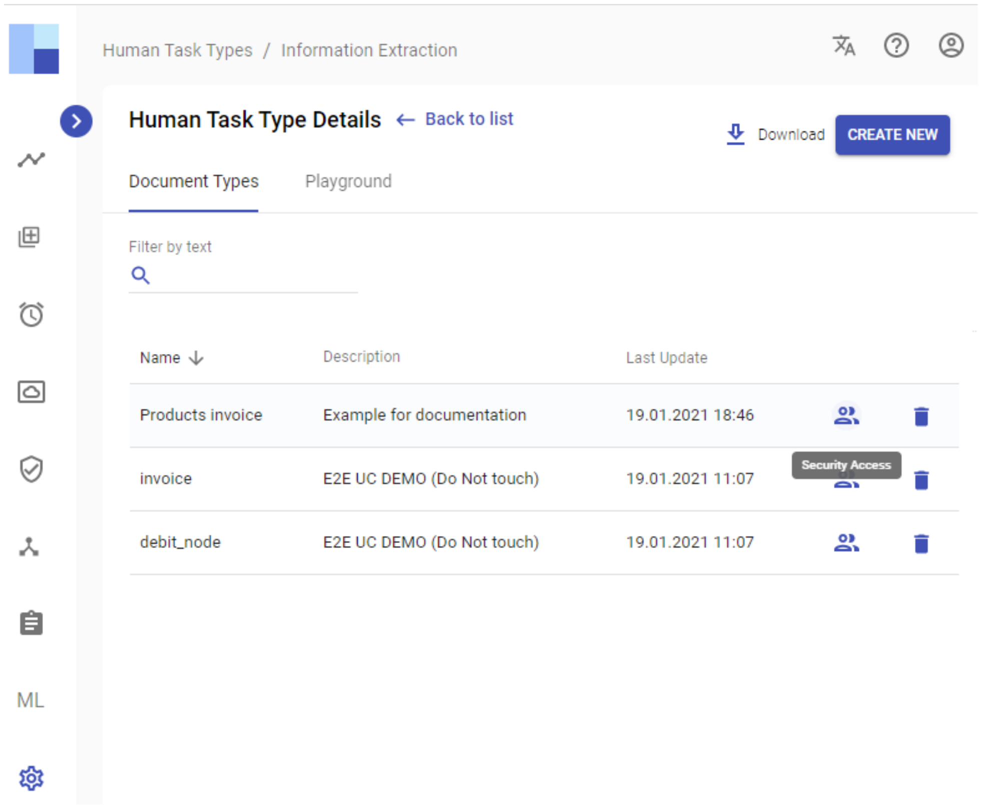
```
{
  "settings": {
    "categories": [
      {
        "name": "Invoice Number",
        "multiple": false
      },
      {
        "name": "Product Name",
        "multiple": true
      },
      {
        "name": "Product Price",
        "multiple": true
      },
      {
        "name": "Total Amount",
        "multiple": false
      }
    ]
  },
  "readonly": true,
  "inputJson": {
    "images": [
      {
        "content":
"data:jpeg;base64,/9j/4AAQSkZJRgABAQABwgHCAAD/4gogSUNDX1BST0ZJTEUAAQEAAAoQA
AAAAAIQAABtbnRyUkdCIFhZWiAAAAAAAAAAAAAAAABhY3NwQVBQTAAAAAAAAAAAAAAAAAAAAAA
```

```
{}
```

Access restriction for Document Type

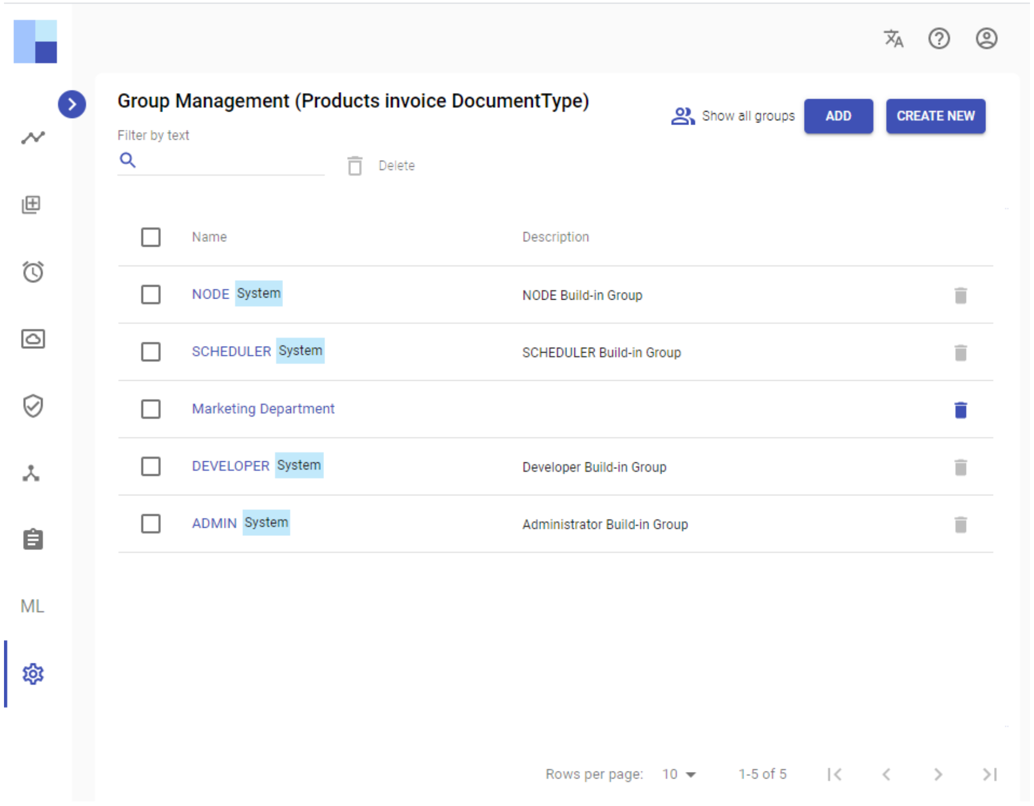
Each Document Type can be restricted to be visible only for some group of people. It's useful when you need to let different teams work only with document types they in charge of. E.g. your Insurance Department should have access to Insurance Claims document type while Marketing Department should be able to see only Advertising Invoices document type.

There's the special "Security Access" link on your Document Type to configure the list of groups which have access to that Document Type:



Name	Description	Last Update
Products invoice	Example for documentation	19.01.2021 18:46
invoice	E2E UC DEMO (Do Not touch)	19.01.2021 11:07
debit_node	E2E UC DEMO (Do Not touch)	19.01.2021 11:07

It opens the Group Management for that particular Document Type where you can manage accesses:

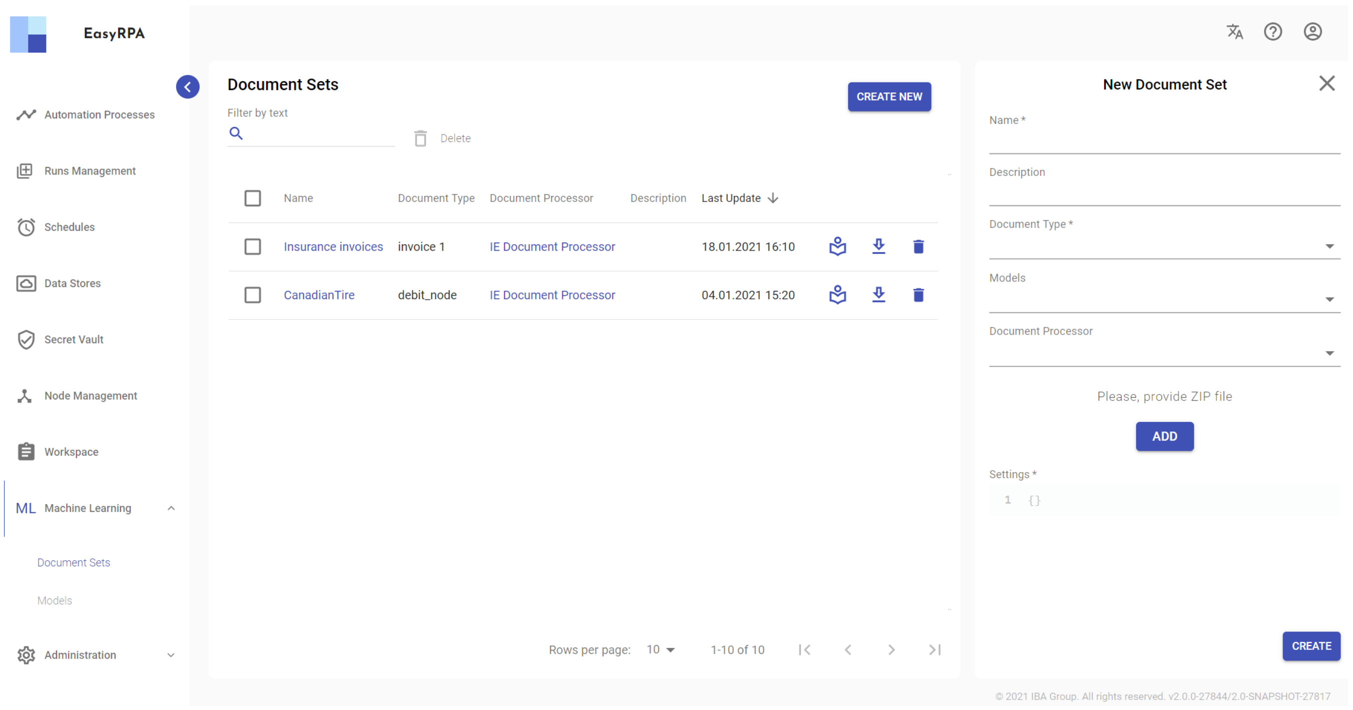


	Name	Description
☐	NODE System	NODE Build-in Group
☐	SCHEDULER System	SCHEDULER Build-in Group
☐	Marketing Department	
☐	DEVELOPER System	Developer Build-in Group
☐	ADMIN System	Administrator Build-in Group

Rows per page: 10 1-5 of 5

(v. 2.2) Step 2. Collect and prepare training set (IE)

To prepare the Training Set which is used to train new ML model, you need to create new **Document Set** entity. Click **Machine Learning** category **Document Sets** **Create New**:



Document Sets

Filter by text

	Name	Document Type	Document Processor	Description	Last Update
☐	Insurance invoices	invoice 1	IE Document Processor		18.01.2021 16:10
☐	CanadianTire	debit_node	IE Document Processor		04.01.2021 15:20

Rows per page: 10 1-10 of 10

New Document Set

Name *

Description

Document Type *

Models

Document Processor

Please, provide ZIP file

Settings *

```
{}
```

© 2021 IBA Group. All rights reserved. v2.0.0-27844/2.0-SNAPSHOT-27817

Then training set preparation contains the following steps:

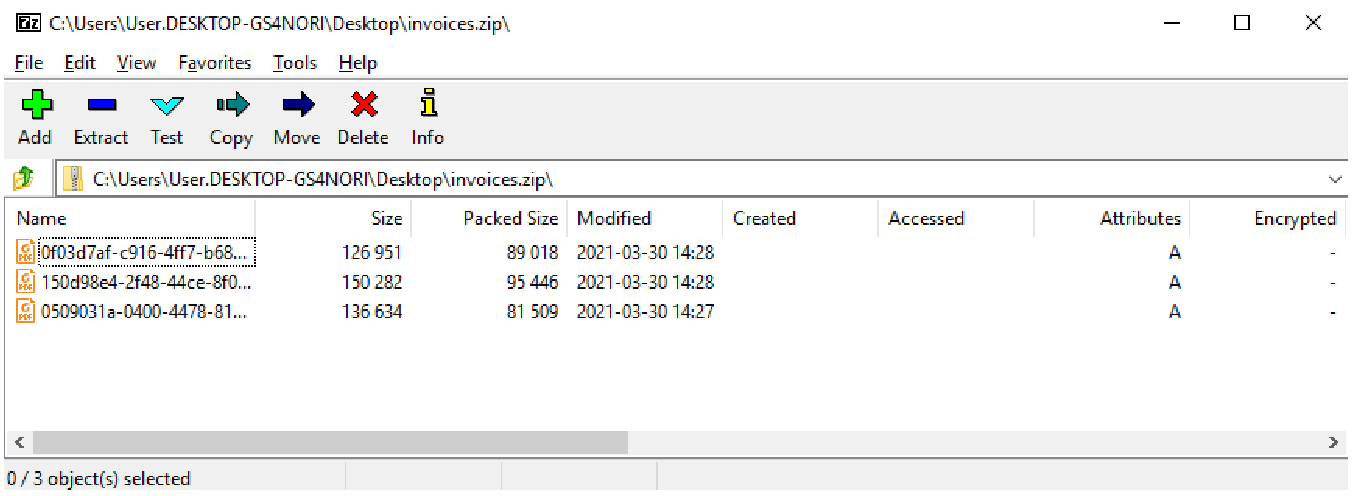
- Provide ZIP file
- Select Document Type
- Select Document Processor
- Generate Document Input
- Move documents to Human Task and tag them

Provide ZIP file

The main information of your training set is your documents. You need to put inside ZIP archive the set of your documents you want to use to train new ML model.

> By default EasyRPA supports processing of **PDF** and **Images** documents to train a model.

Your archive should looks like the following:


C:\Users\User.DESKTOP-GS4NORI\Desktop\invoices.zip\
File Edit View Favorites Tools Help
Add Extract Test Copy Move Delete Info
C:\Users\User.DESKTOP-GS4NORI\Desktop\invoices.zip\

Name	Size	Packed Size	Modified	Created	Accessed	Attributes	Encrypted
0f03d7af-c916-4ff7-b68...	126 951	89 018	2021-03-30 14:28			A	-
150d98e4-2f48-44ce-8f0...	150 282	95 446	2021-03-30 14:28			A	-
0509031a-0400-4478-81...	136 634	81 509	2021-03-30 14:27			A	-

0 / 3 object(s) selected

Select Document Type

Select Document Type created on the previous step. This Document Type contains information of how to display your documents on Human Task for tagging and which output fields should be extract during the tagging and Machine Learning training.

Select Document Processor

Document Processor is a special process which includes preparation steps to display your documents on the Human Task.

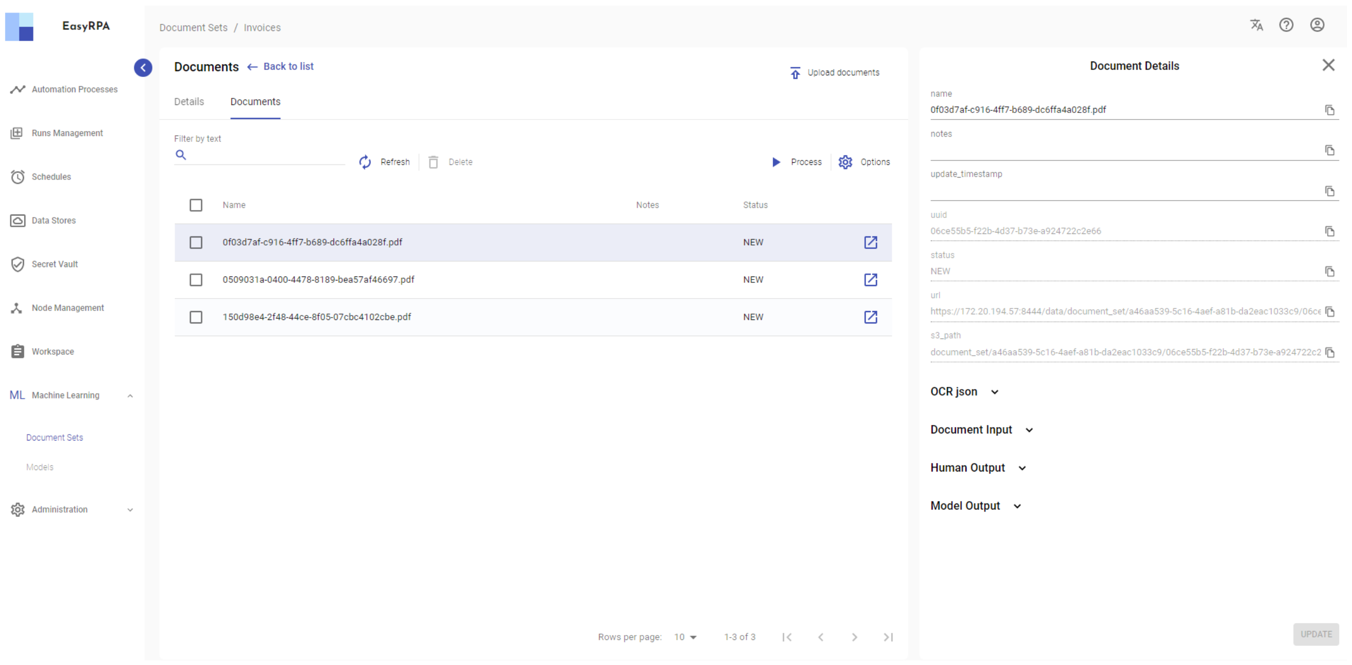
To process PDFs and Images for Information Extraction, EasyRPA provides "**IE Document Processor**" which should be selected in our case. This processor includes:

- converting PDFs into images format
- images improvements using ImageMagick scripts
- sending images to OCR
- sending documents with OCR response to Human Task
- process Human Task response to convert results into Machine Learning input format

After you provided ZIP archive, selected Document Type and Document Processor, you can create document set.

All documents from your Document Set will be uploaded to File Storage.

If you open created Document Set, you can see the details of each document inside:



Name	Notes	Status
0f03d7af-c916-4ff7-b689-dc6ffa4a028f.pdf		NEW
0509031a-0400-4478-8189-bea57af46697.pdf		NEW
150d98e4-2f48-44ce-8f05-07cbc4102cbe.pdf		NEW

Document Details

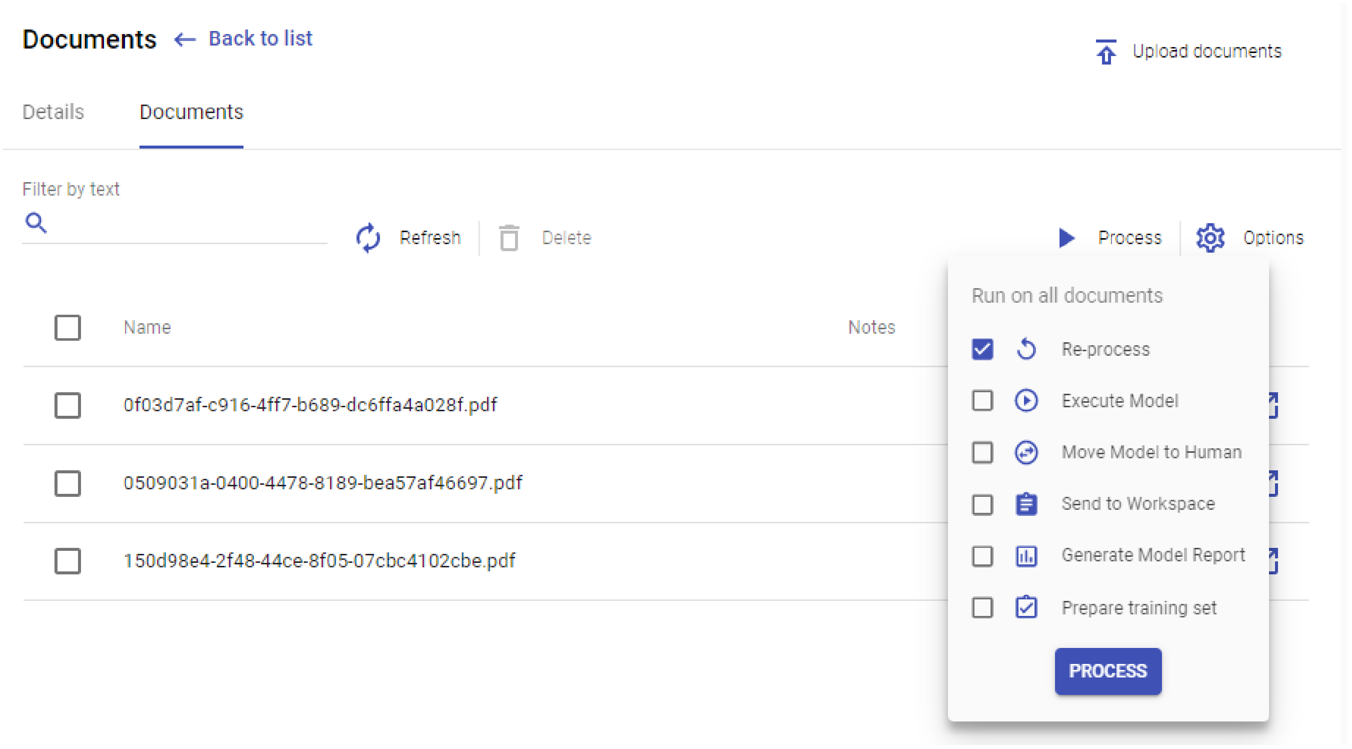
name: 0f03d7af-c916-4ff7-b689-dc6ffa4a028f.pdf
notes:
update_timestamp:
uuid: 06ce55b5-f22b-4d37-b73e-a924722c2e66
status: NEW
url: https://172.20.194.57:8444/data/document_set/a46aa539-5c16-4aef-a81b-da2eac1033c9/06ce
s3_path: document_set/a46aa539-5c16-4aef-a81b-da2eac1033c9/06ce55b5-f22b-4d37-b73e-a924722c2

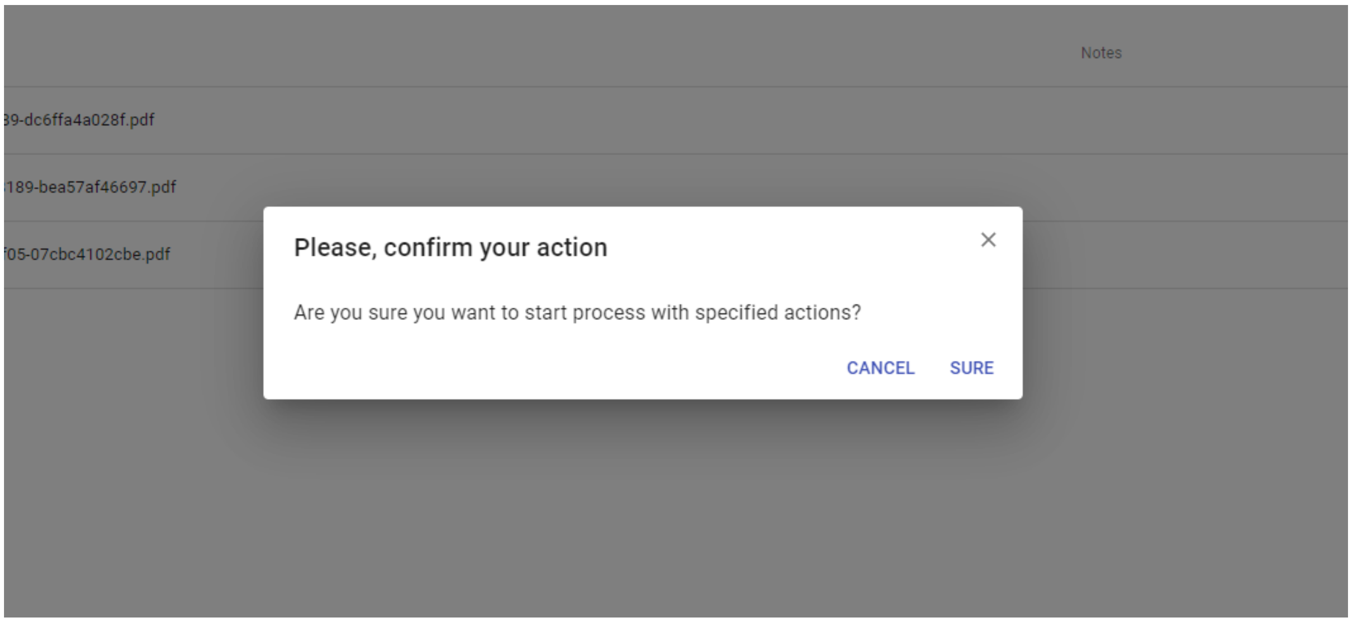
OCR json
Document Input
Human Output
Model Output

As this Document Set is new, **Document Input**, **Human Output** and **Model Output** details are empty.

Generate Document Input

Document Input is an input data which comes as an input for Human Tasks. As mentioned before, Document Processor is responsible for preparation of that input. To trigger Document Processor and to prepare Document Input, you should **select "Re-process" action** from the action menu and then confirm the in appeared confirmation dialog:





Notes
39-dc6ffa4a028f.pdf
189-bea57af46697.pdf
f05-07cbc4102cbe.pdf

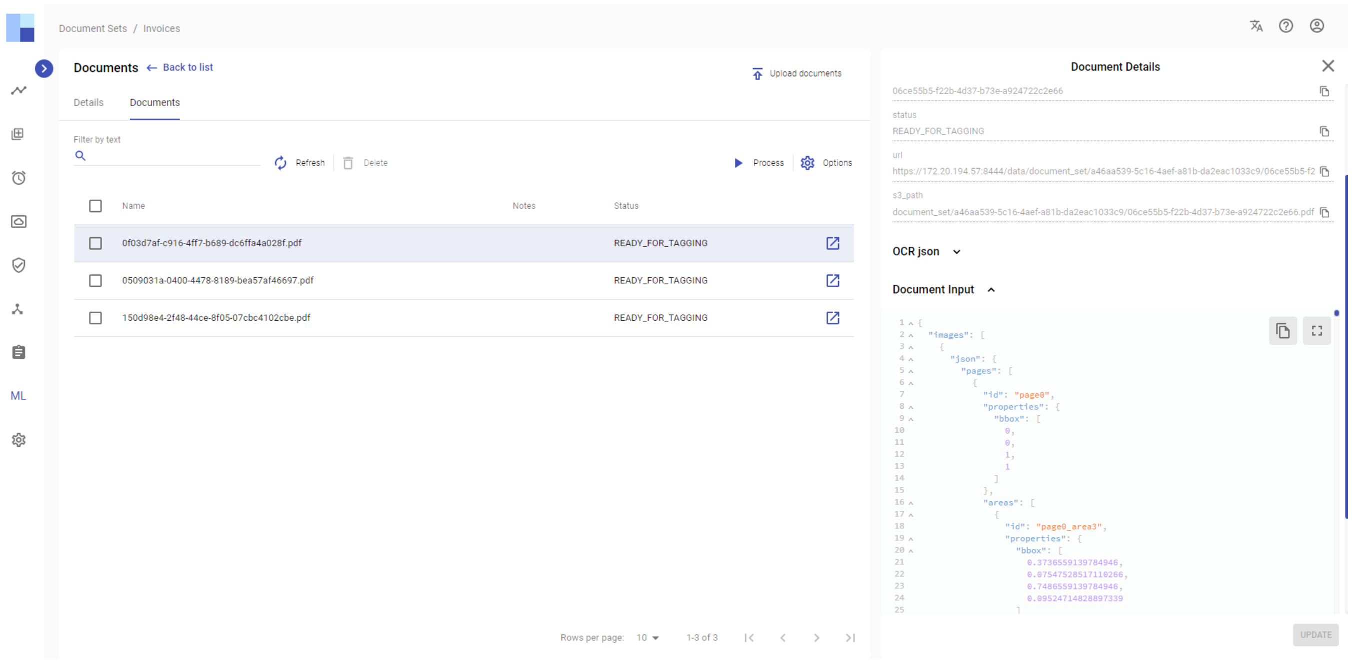
Please, confirm your action

Are you sure you want to start process with specified actions?

Document Processor will start working. You can monitor it in Runs Management.

Wait until the status of all documents becomes "**READY_FOR_TAGGING**"

As the result you will see that "**Document input**" details are generated where you will find the input data from Human Task. This data contains the result of OCR in an appropriate format:



Document Sets / Invoices

Documents ← Back to list

	Name	Notes	Status
☐	0f03d7af-c916-4ff7-b689-dc6ffa4a028f.pdf		READY_FOR_TAGGING
☐	0509031a-0400-4478-8189-bea57af46697.pdf		READY_FOR_TAGGING
☐	150d98e4-2f48-44ce-8f05-07cbc4102cbe.pdf		READY_FOR_TAGGING

Document Details

06ce55b5-f22b-4d37-b73e-a924722c2e66

status
READY_FOR_TAGGING

url
https://172.20.194.57:8444/data/document_set/a46aa539-5c16-4aef-a81b-da2eac1033c9/06ce55b5-f2

s3_path
document_set/a46aa539-5c16-4aef-a81b-da2eac1033c9/06ce55b5-f22b-4d37-b73e-a924722c2e66.pdf

OCR json

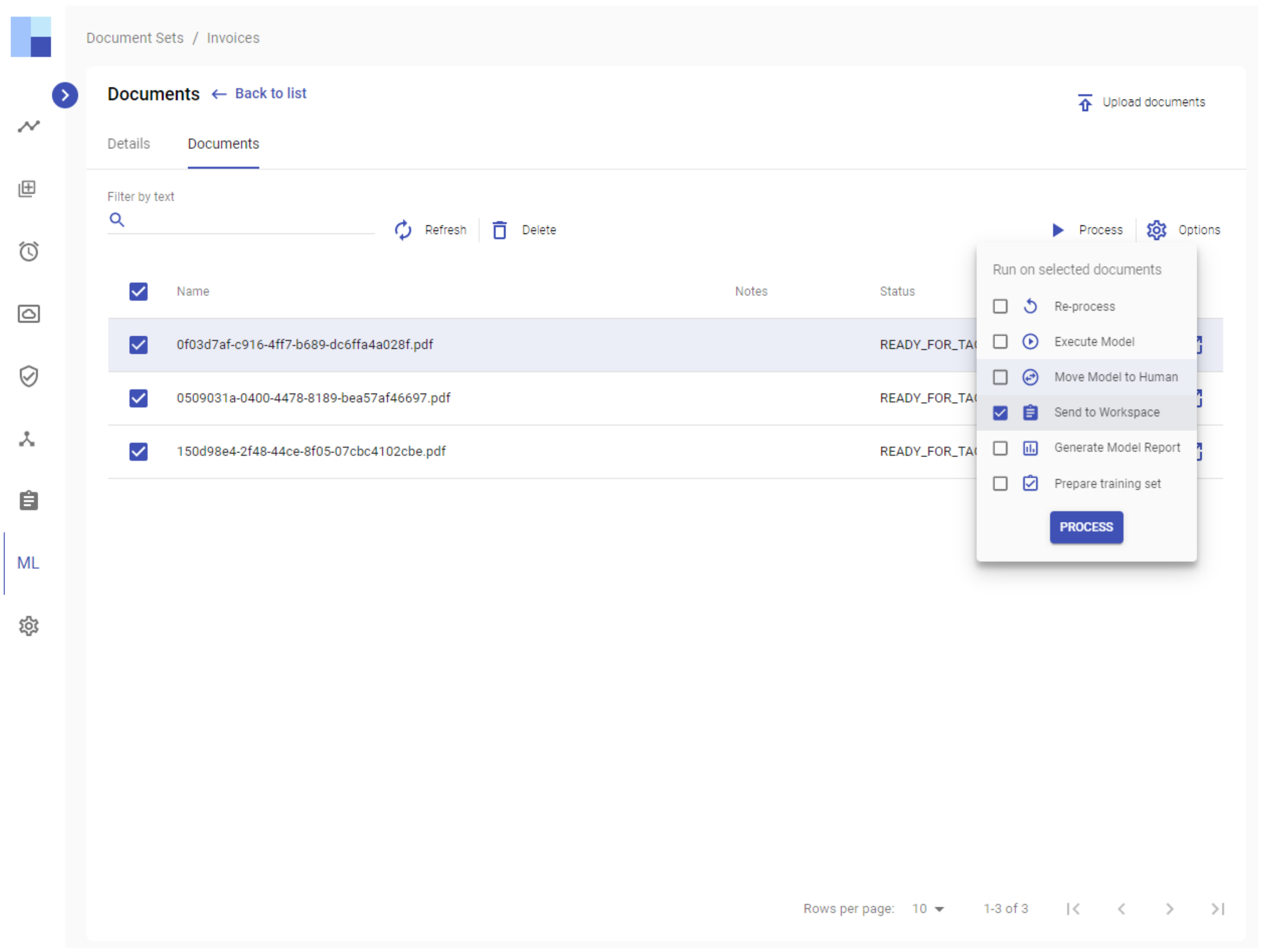
Document Input

```
{
  "images": [
    {
      "json": {
        "pages": [
          {
            "id": "page0",
            "properties": {
              "bbox": [
                0,
                0,
                1,
                1
              ]
            },
            "areas": [
              {
                "id": "page0_area3",
                "properties": {
                  "bbox": [
                    0.3736559139784946,
                    0.07547528517110266,
                    0.7486559139784946,
                    0.09524714828897339
                  ]
```

Move documents to Human Task and tag them

The next step we need to tag the documents. This result of tagging will be used by ML algorithms to train a model.

To move your Document Set to Workspace and to let workers to tag them via Human Task you need to select "**Send to Workspace**" action:



Document Sets / Invoices

Documents ← Back to list

Upload documents

Details | Documents

Filter by text

Refresh | Delete | Process | Options

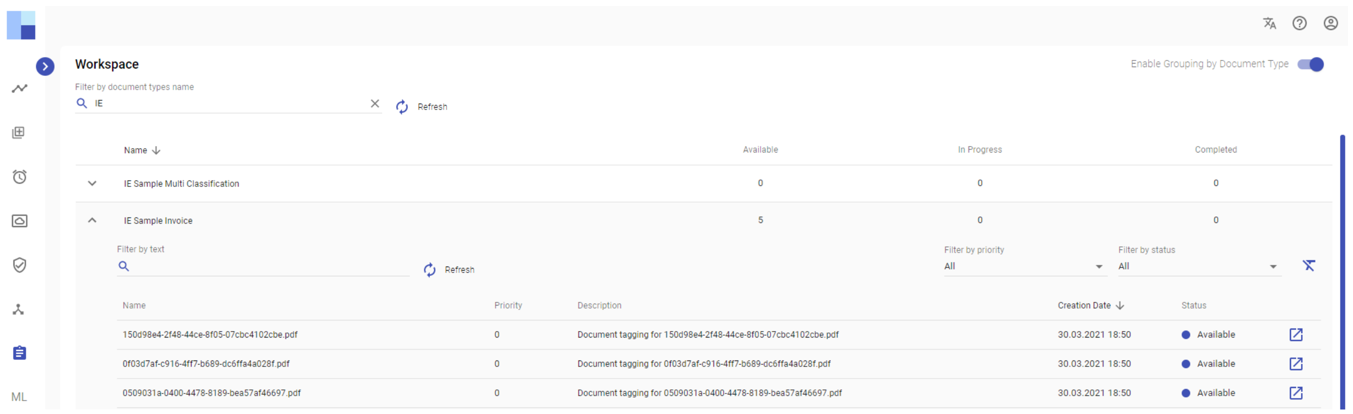
Run on selected documents
- [ ] Re-process
- [ ] Execute Model
- [ ] Move Model to Human
- [✓] Send to Workspace
- [ ] Generate Model Report
- [ ] Prepare training set

PROCESS

✓	Name	Notes	Status
✓	0f03d7af-c916-4ff7-b689-dc6ffa4a028f.pdf		READY_FOR_TA
✓	0509031a-0400-4478-8189-bea57af46697.pdf		READY_FOR_TA
✓	150d98e4-2f48-44ce-8f05-07cbc4102cbe.pdf		READY_FOR_TA

Rows per page: 10 1-3 of 3

Documents should be appeared in Workspace under corresponding Document Type. You can click button to start tagging it:



Workspace

Enable Grouping by Document Type

Filter by document types name: IE

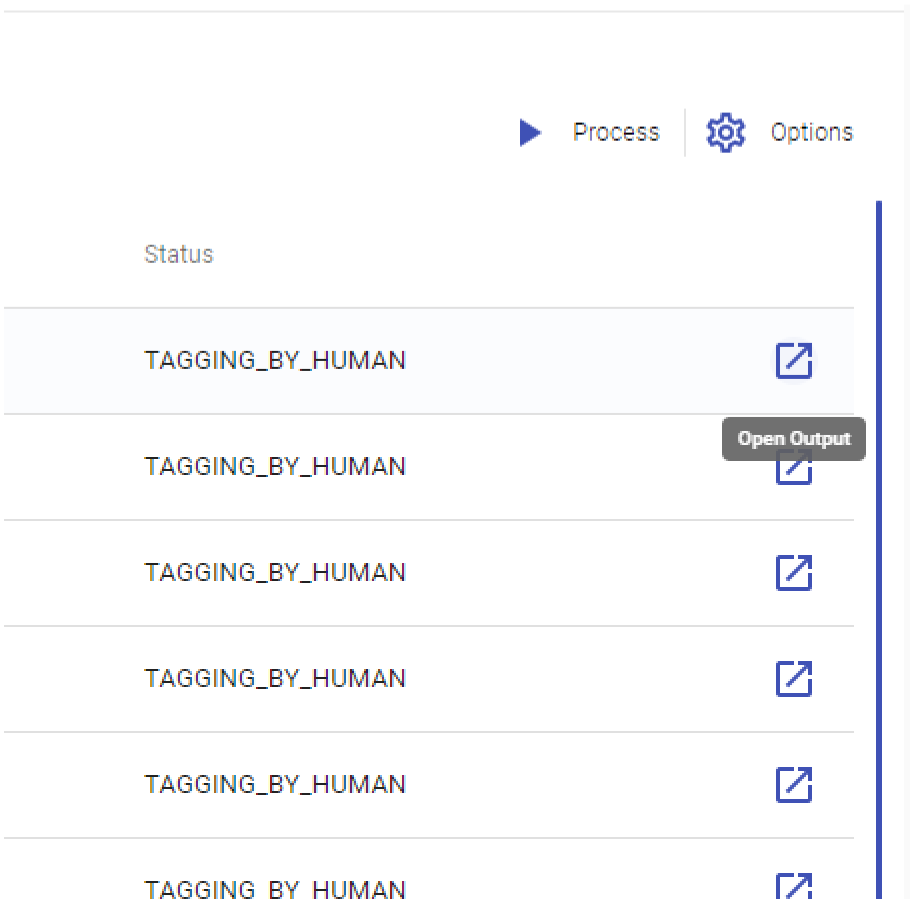
Refresh

	Name ↓	Available	In Progress	Completed
⌄	IE Sample Multi Classification	0	0	0
⌃	IE Sample Invoice	5	0	0

Filter by text | Refresh | Filter by priority: All | Filter by status: All

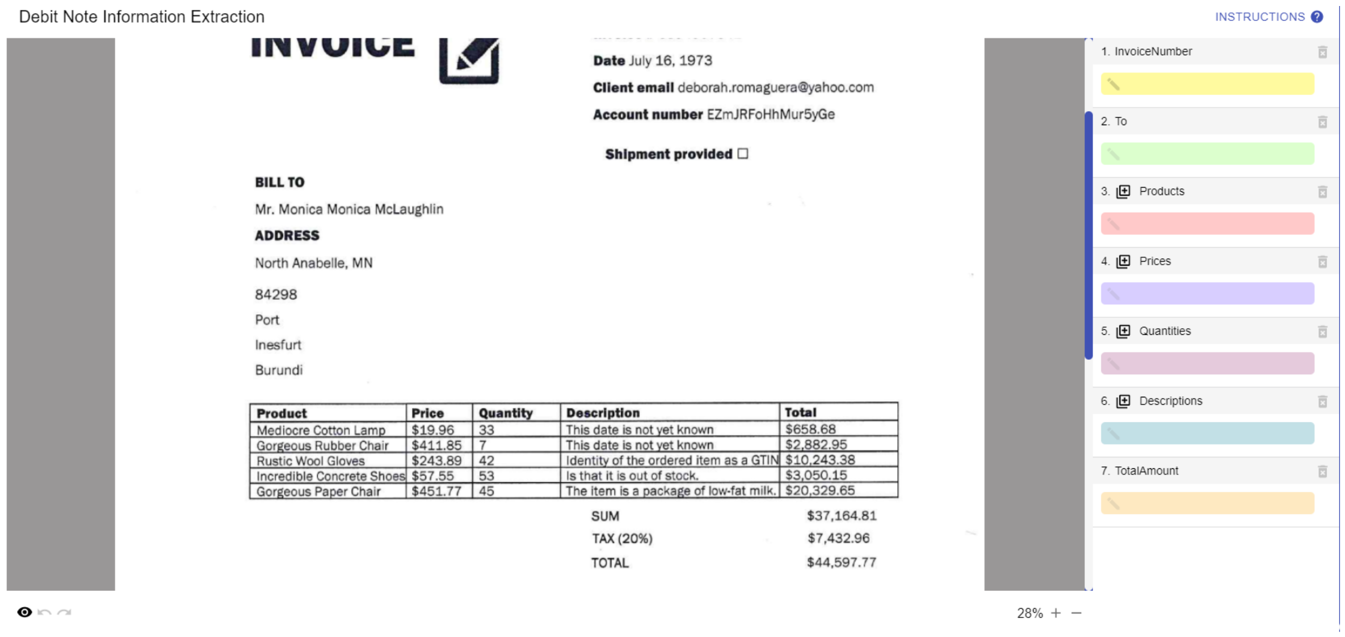
Name	Priority	Description	Creation Date ↓	Status
150d98e4-2f48-44ce-8f05-07cbc4102cbe.pdf	0	Document tagging for 150d98e4-2f48-44ce-8f05-07cbc4102cbe.pdf	30.03.2021 18:50	● Available
0f03d7af-c916-4ff7-b689-dc6ffa4a028f.pdf	0	Document tagging for 0f03d7af-c916-4ff7-b689-dc6ffa4a028f.pdf	30.03.2021 18:50	● Available
0509031a-0400-4478-8189-bea57af46697.pdf	0	Document tagging for 0509031a-0400-4478-8189-bea57af46697.pdf	30.03.2021 18:50	● Available

** There's also an option to tag document without sending it to the Workspace. So you can click "Open Output" button next to the status label inside you Document Set details:*



Status
TAGGING_BY_HUMAN
TAGGING_BY_HUMAN
TAGGING_BY_HUMAN
TAGGING_BY_HUMAN
TAGGING_BY_HUMAN
TAGGING_BY_HUMAN

You can start tagging document:



Debit Note Information Extraction

INVOICE

Date July 16, 1973

Client email deborah.romaguera@yahoo.com

Account number EZmJRFoHhMur5yGe

Shipment provided ☐

BILL TO

Mr. Monica Monica McLaughlin

ADDRESS

North Anabelle, MN

84298

Port

Inesfurt

Burundi

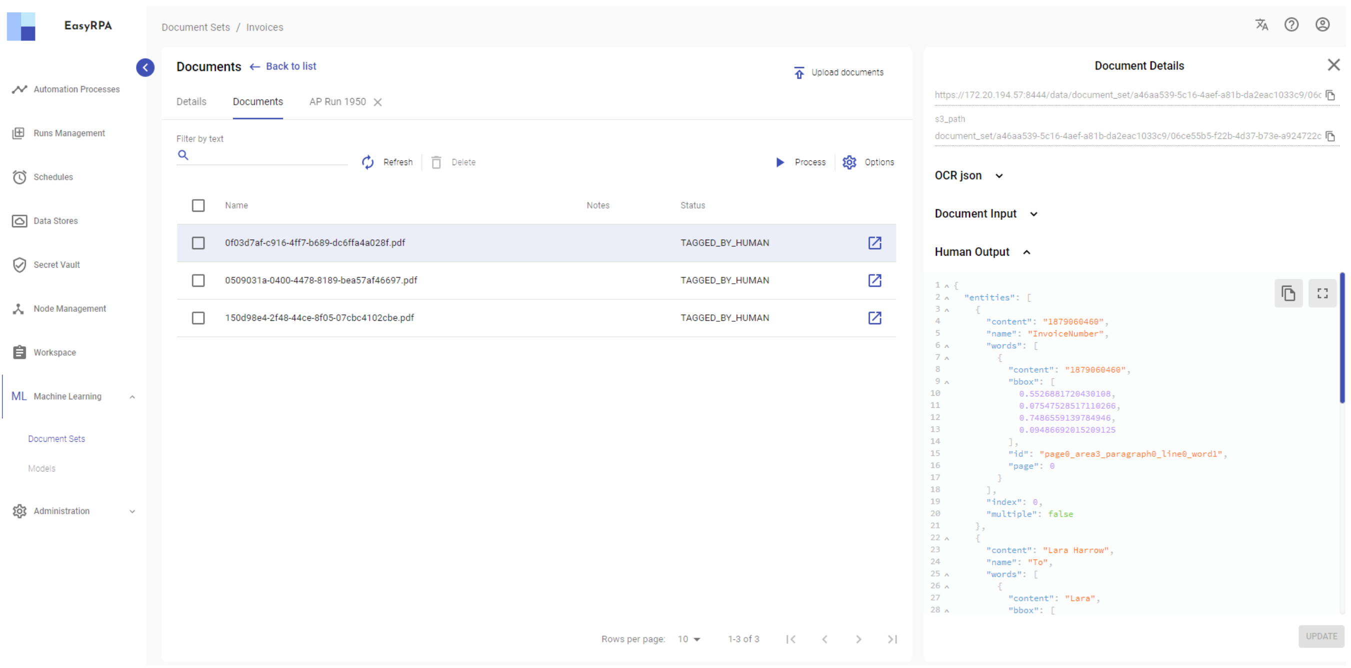
Product	Price	Quantity	Description	Total
Mediocre Cotton Lamp	$19.96	33	This date is not yet known	$658.68
Gorgeous Rubber Chair	$411.85	7	This date is not yet known	$2,882.95
Rustic Wool Gloves	$243.89	42	Identity of the ordered item as a GTIN	$10,243.38
Incredible Concrete Shoes	$57.55	53	Is that it is out of stock.	$3,050.15
Gorgeous Paper Chair	$451.77	45	The item is a package of low-fat milk.	$20,329.65

SUM $37,164.81

TAX (20%) $7,432.96

TOTAL $44,597.77

After document is tagged, "**Human Output**" details are generated for that particular document:



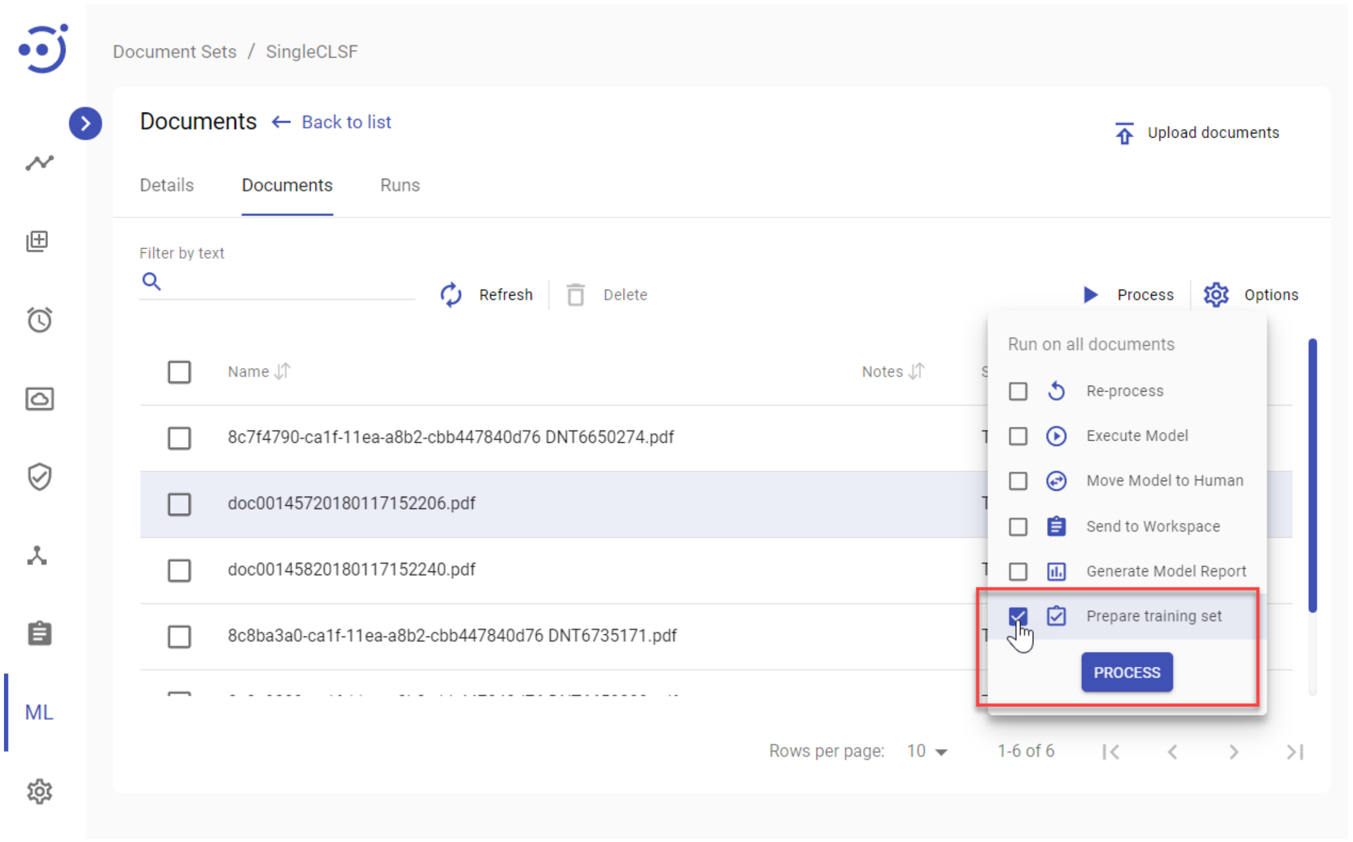
	Name	Notes	Status
☐	0f03d7af-c916-4ff7-b689-dc6ffa4a028f.pdf		TAGGED_BY_HUMAN
☐	0509031a-0400-4478-8189-bea57af46697.pdf		TAGGED_BY_HUMAN
☐	150d98e4-2f48-44ce-8f05-07cbc4102cbe.pdf		TAGGED_BY_HUMAN

https://172.20.194.57:8444/data/document_set/a46aa539-5c16-4aef-a81b-da2eac1033c9/06c

s3_path
document_set/a46aa539-5c16-4aef-a81b-da2eac1033c9/06ce55b5-f22b-4d37-b73e-a924722c

```
{
  "entities": [
    {
      "content": "1879060460",
      "name": "InvoiceNumber",
      "words": [
        {
          "content": "1879060460",
          "bbox": [
            0.5526881720430108,
            0.07547528517110266,
            0.7486559139784946,
            0.09486692015209125
          ],
          "id": "page0_area3_paragraph0_line0_word1",
          "page": 0
        }
      ],
      "index": 0,
      "multiple": false
    },
    {
      "content": "Lara Harrow",
      "name": "To",
      "words": [
        {
          "content": "Lara",
          "bbox": [
```

To finish Training set preparation click **Process Prepare training set** and confirm your action with **Process** button:



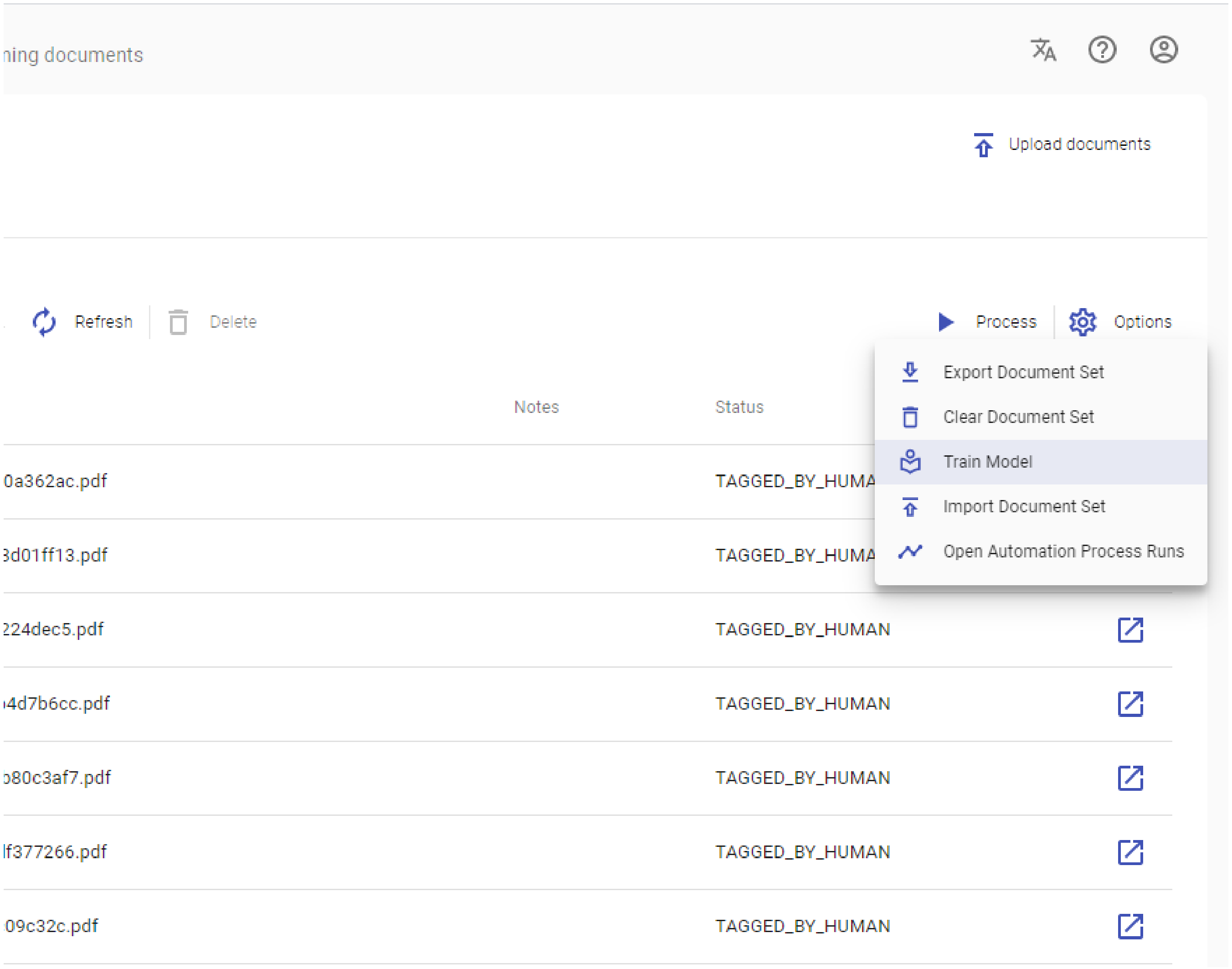
	Name	Notes
☐	8c7f4790-ca1f-11ea-a8b2-cbb447840d76 DNT6650274.pdf	
☐	doc00145720180117152206.pdf	
☐	doc00145820180117152240.pdf	
☐	8c8ba3a0-ca1f-11ea-a8b2-cbb447840d76 DNT6735171.pdf	

After completing these actions you can proceed to model training.

(v. 2.2) Step 3. Train ML model (IE)

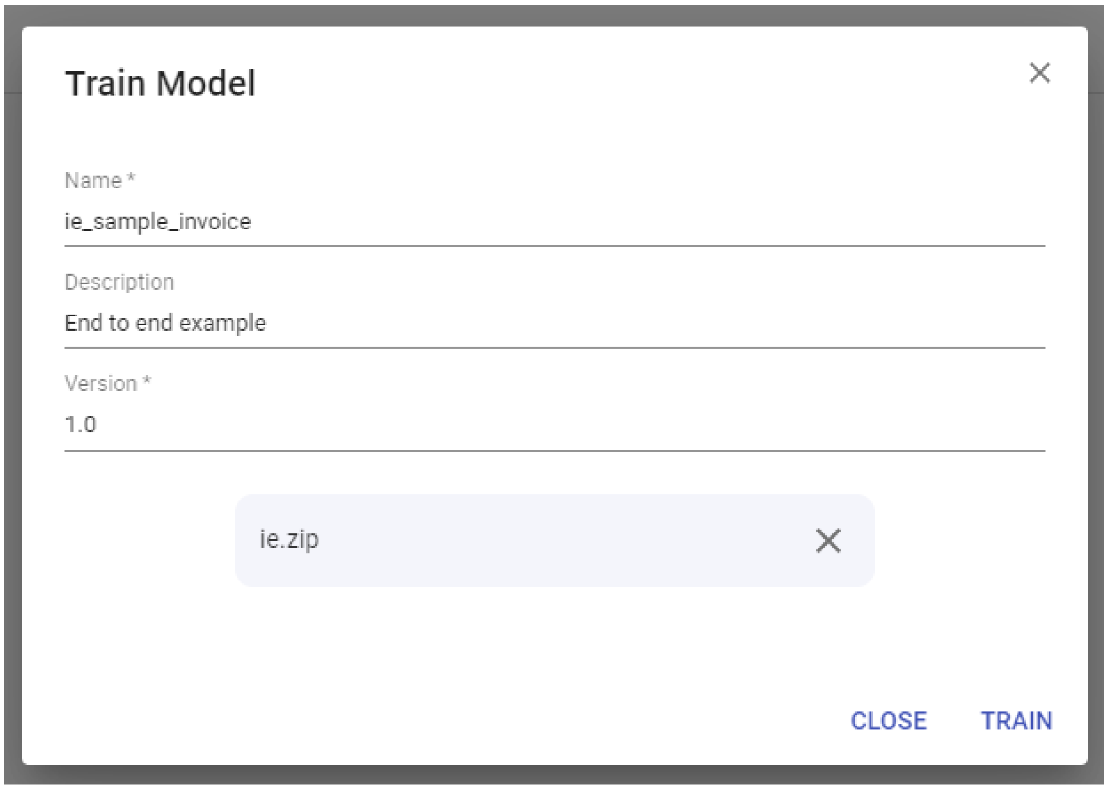
After you prepared training set and tag all the documents, you can train your ML model.

Inside Document Set select **Options** **Train Model**



	Notes	Status
0a362ac.pdf		TAGGED_BY_HUMA
3d01ff13.pdf		TAGGED_BY_HUMA
224dec5.pdf		TAGGED_BY_HUMAN
4d7b6cc.pdf		TAGGED_BY_HUMAN
080c3af7.pdf		TAGGED_BY_HUMAN
lf377266.pdf		TAGGED_BY_HUMAN
09c32c.pdf		TAGGED_BY_HUMAN

Following dialog appears:


Train Model

Name *
ie_sample_invoice

Description
End to end example

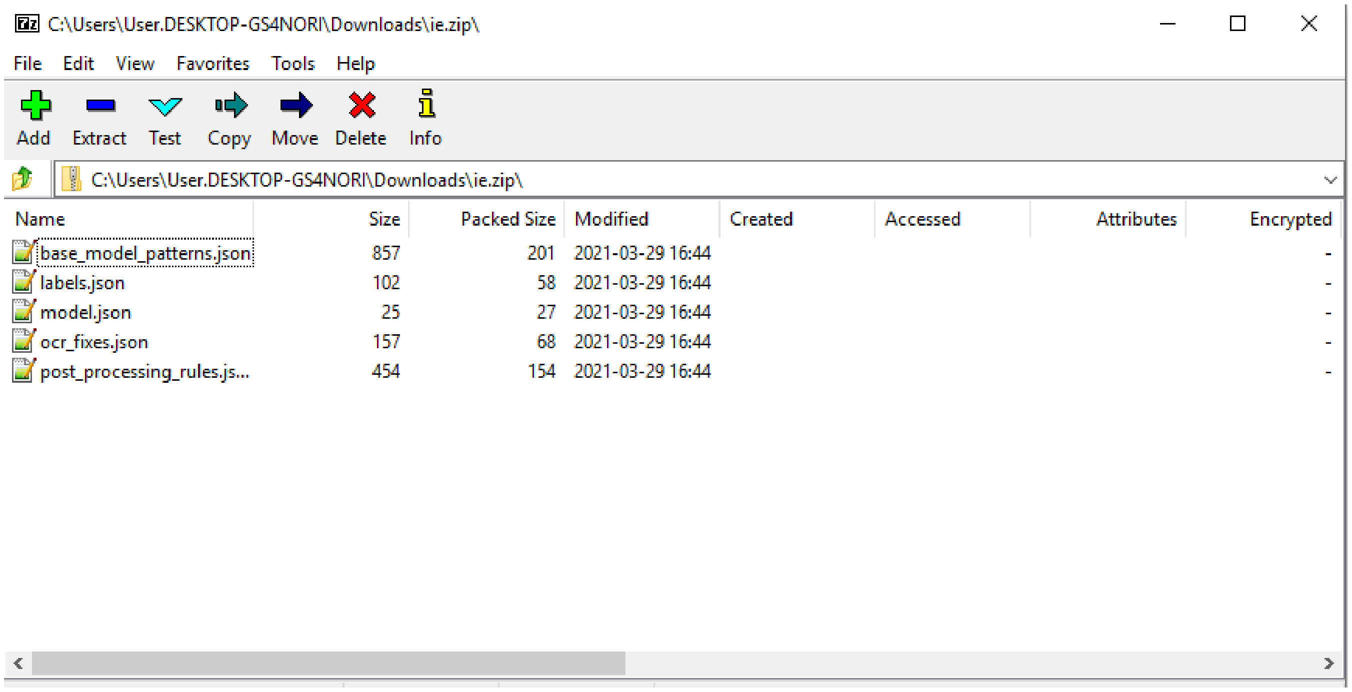
Version *
1.0

ie.zip

You need to specify:

- Model name
- Model description
- Model version
- Attach ZIP file with configurations.

ZIP file with configurations should contains Model Training Configuration Files


C:\Users\User.DESKTOP-GS4NORI\Downloads\ie.zip\

File Edit View Favorites Tools Help
Add Extract Test Copy Move Delete Info

Name	Size	Packed Size	Modified	Created	Accessed	Attributes	Encrypted
base_model_patterns.json	857	201	2021-03-29 16:44				-
labels.json	102	58	2021-03-29 16:44				-
model.json	25	27	2021-03-29 16:44				-
ocr_fixes.json	157	68	2021-03-29 16:44				-
post_processing_rules.js...	454	154	2021-03-29 16:44				-

model.json configuration file

It has the following structure:

model.json

```
{
  "model_type": "ie"
}
```

- **model_type** (string) - for Information Extraction model it should be always "**textcat**" value

labels.json configuration file

It has the following structure:

model.json

```
{
  "InvoiceNumber": false,
  "BillTo": false,
  "Subtotal": false,
  "Tax": false,
  "Total": false
}
```

post_processing_rules.json configuration file

It has the following structure:

model.json

```
[
  {
    "label": "InvoiceNumber",
    "regex": [
      "Invoice # (...\\d{10})"
    ]
  },
    {
    "label": "BillTo",
    "regex": [
      "BILL TO ([ a-zA-Z]+)"
    ]
  },
    {
    "label": "Subtotal",
    "regex": [
      "SUM (...[$,.\\d]+)"
    ]
  },
    {
    "label": "Tax",
    "regex": [
      "TAX (...[$,.\\d]+)"
    ]
  },
    {
    "label": "Total",
    "regex": [
      "TOTAL (...[$,.\\d]+)"
    ]
  }
]
```

base_model_patterns.json configuration file

It has the following structure:

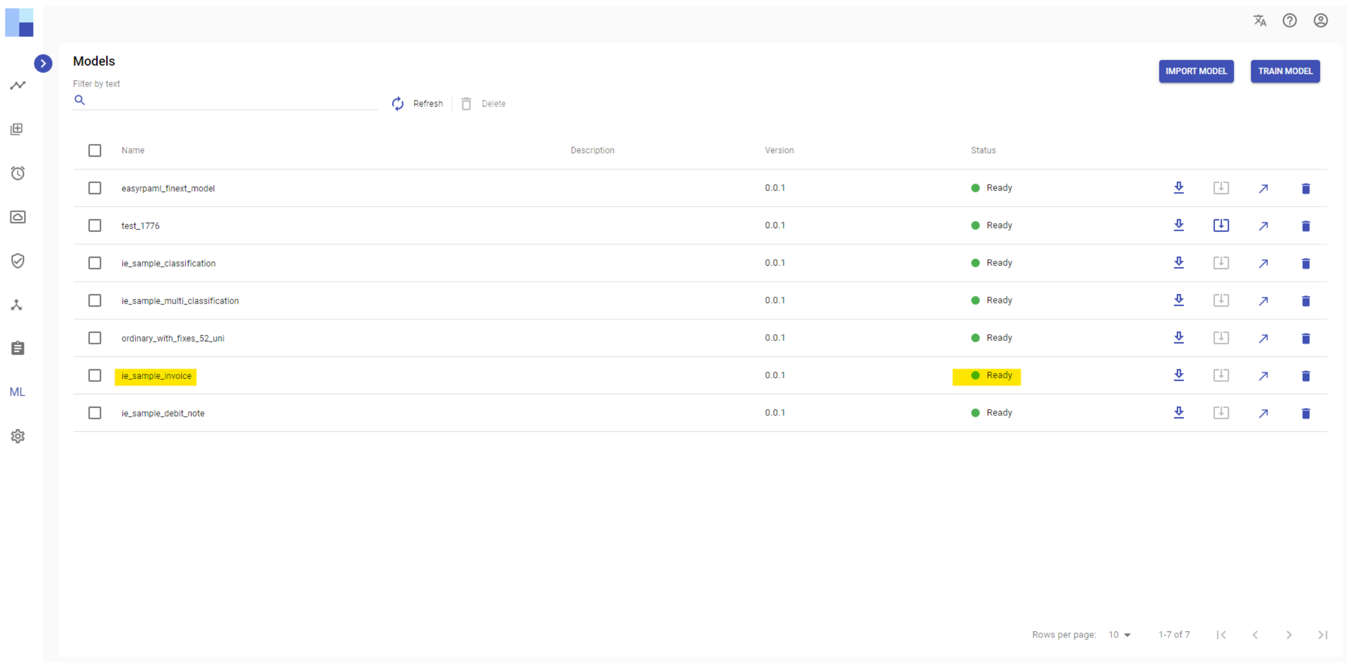
ocr_fixes.json

```
[
  {
    "label": "kwInvoiceNumber",
    "id": "kwInvoiceNumber",
    "pattern": [
      { "TEXT": "Invoice" },
      { "TEXT": { "REGEX": "^Invoice.*$" } },
      { "TEXT": "#" }
    ]
  },
    {
    "label": "kwBillTo",
    "id": "kwBillTo",
    "pattern": [
      { "TEXT": "BILL TO" },
      { "TEXT": { "REGEX": "^BILL TO.*$" } }
    ]
  },
    {
    "label": "kwSubtotal",
    "id": "kwSubtotal",
    "pattern": [
      { "TEXT": "SUM" },
      { "TEXT": { "REGEX": "^SUM.*$" } },
      { "TEXT": "$" }
    ]
  },
    {
    "label": "kwTax",
    "id": "kwTax",
    "pattern": [
      { "TEXT": "TAX" },
      { "TEXT": { "REGEX": "^TAX.*$" } },
      { "TEXT": "$" }
    ]
  },
    {
    "label": "kwTotal",
    "id": "kwTotal",
    "pattern": [
      { "TEXT": "TOTAL" },
      { "TEXT": { "REGEX": "^TOTAL.*$" } },
      { "TEXT": "$" }
    ]
  }
]
```

ocr_fixes.json configuration file

Now it is not used in the model configuration and has been moved to the OCR config section.

After you click "TRAIN" button on confirmation dialog - wait until model training is finished and your model becomes with status "Ready". You can find the details on **Machine Learning Models** page:


Models

Filter by text

Refresh Delete

IMPORT MODEL TRAIN MODEL

Name	Description	Version	Status
easyrpaml_finext_model		0.0.1	● Ready
test_1776		0.0.1	● Ready
ie_sample_classification		0.0.1	● Ready
ie_sample_multi_classification		0.0.1	● Ready
ordinary_with_fixes_52_uni		0.0.1	● Ready
ie_sample_invoice		0.0.1	● Ready
ie_sample_debit_note		0.0.1	● Ready

Rows per page: 10 1-7 of 7

(v. 2.2) Develop Automation Process (IE)

(v. 2.2) Step 1. Prepare input documents (IE)

Retrieve documents from the data source

Basically, the Information Extraction process begins with the document retrieving step. Usually, the data source is a mailbox that you need to scan to find emails using some search criteria, and attachments from those emails are your input data.

Below is the list of data source examples and suggestions on how to work with them.

Data Source	Suggestions
Emails in some mailbox	Depends on the protocol you can access the mailbox you need to use different libraries to scan it. The most popular 2 protocols are: • IMAP • Exchange EasyRPA provides an Email Client utility that covers both protocols and can be used for scanning and sending emails. You can scan emails using necessary search terms like some keyword in the subject, emails from a specific sender, emails in the date range, etc.
Files from Shared Network Folder	Sometimes you need to scan a specific folder in the network drive for some files to appear. There're 2 ways you can access the network folders: 1. Using the Samba protocol client for Java (e.g. https://github.com/AgNO3/jcifs-ng). 2. You can map a network drive using the Windows feature (https://support.microsoft.com/en-us/windows/map-a-network-drive-in-windows-10-29ce55d1-34e3-a7e2-4801-131475f9557d) so after that, you can access the network folder in the same way as a local file system.
Files from EasyRPA file storage	It's a good option if you can agree with the business process operators to put target documents for processing into EasyRPA file storage. Then you can scan it using the existing Storage Manager
Files from FTP	It's also possible that you need to scan the FTP server to retrieve incoming documents. So you need to use some FTP client library for Java (e.g. Apache Commons Net). Also pay attention, that sometimes FTP servers have a simple Web UI interface, so you can access the files using common HTTP Get requests.

Upload documents to EasyRPA File Storage

All documents you retrieve from the previous step you should upload into File Storage, because further in OCR and ML step you need to work with URLs to documents, not documents by itself.

You can easily upload data into File Storage using embedded Storage Manager.

Code example

In our step-by-step example we're going to implement EasyRPA Storage scanning for new incoming invoices.

So Automation Process (AP) class will looks like the following:

InvoiceProcessingSample.java

```
package eu.ibagroup.samples.ie;
```

```
import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModule;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import eu.ibagroup.samples.ie.task.GetIncomingInvoices;
import lombok.extern.slf4j.Slf4j;

@ApModuleEntry(name = "InvoiceProcessingSample")
@Slf4j
public class InvoiceProcessingSample extends ApModule {

    public TaskOutput run() throws Exception {
        TaskOutput incomingInvoicesResult = execute(getInput(), GetIncomingInvoices.class).get();
        return doNotCareOfResult();
    }
}
```

And Task by itself which scan all PDF files from specific folder in file storage and then move it to different working folder:

GetIncomingInvoices.java

```
package eu.ibagroup.samples.ie.task;

import eu.ibagroup.easyrpa.engine.annotation.AfterInit;
import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.annotation.InputToOutput;
import eu.ibagroup.easyrpa.engine.annotation.Output;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.utils.storage.StorageManager;
import eu.ibagroup.samples.ie.ExportConstants;
import eu.ibagroup.samples.ie.PropertyConstants;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.io.FilenameUtils;

import javax.inject.Inject;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

@ApTaskEntry(name = "Get Documents from Storage")
@Slf4j
@InputToOutput
public class GetIncomingInvoices extends ApTask {

    @Inject
    private StorageManager storageManager;

    @Output(ExportConstants.INVOICE_DOCUMENT_S3_PATHS_LIST)
    private List<String> invoicesS3PathsList = new ArrayList<>();

    private String s3Bucket;
    private String s3FolderToScan;

    private String ocrS3WorkBucket;
    private String ocrS3WorkFolder;

    @AfterInit
    public void init() {
        s3Bucket = getConfigurationService().getConfiguration().get(PropertyConstants.S3_BUCKET);
        s3FolderToScan = getConfigurationService().getConfiguration().get(PropertyConstants.S3_FOLDER_TO_SCAN);

        ocrS3WorkBucket = getConfigurationService().getConfiguration().get(PropertyConstants.
OCR_S3_WORK_BUCKET);
        ocrS3WorkFolder = getConfigurationService().getConfiguration().get(PropertyConstants.
OCR_S3_WORK_FOLDER);
    }
    @Override
    public void execute() throws Exception {
        List<String> filesPathList = storageManager.listFiles(s3Bucket, s3FolderToScan, ".*\\.pdf");
        log.info("There're {} document(s) found to process on S3 bucket '{}' in folder '{}'", filesPathList.
size(), s3Bucket, s3FolderToScan);
        for (String s3Path : filesPathList) {
            String newS3Path = moveFile(s3Bucket, s3Path, ocrS3WorkBucket, ocrS3WorkFolder);
            invoicesS3PathsList.add(newS3Path);
        }
    }

    private String moveFile(String s3SourceBucket, String s3SourceFilePath, String s3DestBucket, String
s3DestFolder) throws IOException {
        String fileName = FilenameUtils.getName(s3SourceFilePath);
        String resultS3Path = s3DestFolder + "/" + fileName;
        copyFile(s3SourceBucket, s3SourceFilePath, s3DestBucket, resultS3Path);
        storageManager.deleteFile(s3SourceBucket, s3SourceFilePath);
        return resultS3Path;
    }

    private void copyFile(String s3SourceBucket, String s3SourceFilePath, String s3DestBucket, String
s3DestFilePath) throws IOException {
        try(InputStream fileInputStream = storageManager.getFile(s3SourceBucket, s3SourceFilePath)) {
            storageManager.uploadFile(s3DestBucket, s3DestFilePath, fileInputStream);
        }
    }
}
```

PropertyConstants.java

```
package eu.ibagroup.samples.ie;

public interface PropertyConstants {
    String S3_BUCKET = "s3.bucket";
    String S3_FOLDER_TO_SCAN = "s3.folder_to_scan";

    String OCR_S3_WORK_FOLDER = "ocr.s3_work_folder";
    String OCR_S3_WORK_BUCKET = "ocr.s3_work_bucket";
}
```

ExportConstants.java

```
package eu.ibagroup.samples.ie;

public interface ExportConstants {
    String INVOICE_DOCUMENT_S3_PATHS_LIST = "INVOICE_DOCUMENT_S3_PATHS_LIST";
}
```

apm_run.properties

```
# EasyRPA Client Configuration File
s3.bucket=ie-sample
s3.folder_to_scan=incoming_invoices

ocr.s3_work_bucket=data
ocr.s3_work_folder=incoming_invoices_work_folder
```

(v. 2.2) Step 2. Digitize documents using OCR (IE)

The goal of this step is to convert your document into HTML format, because Manual Task and Machine Learning are use this format to process and display your document content.

If your document is already in HTML format (e.g. you converted incoming Excel file into HTML or your document source was email body) you should **skip this step**.

EasyRPA provides special OCR Task which you can use inside your Automation Process code.

Code example

First of all to get all documents paths from the previous step, which scans File Storage, we add the following code into AP process:

```
List<String> invoiceDocumentS3PathList = incomingInvoicesResult.get(ExportConstants.
INVOICE_DOCUMENT_S3_PATHS_LIST, List.class);
```

Then we need to process each document in parallel. For that we need to use CompletableFuture API. So by iteration through document paths we need to prepare the list of Completable Futures and to complete all at the end:

```
List<CompletableFuture<TaskOutput>> invoiceProcessingExecutions = new ArrayList<>();
if(invoiceDocumentS3PathList.size() > 0) {
    for(String documentS3Path : invoiceDocumentS3PathList) {
        TaskInput documentProcessingInput = new TaskInput(incomingInvoicesResult);
        documentProcessingInput.set(ExportConstants.INVOICE_DOCUMENT_S3_PATH, documentS3Path);

        CompletableFuture<TaskOutput> singleInvoiceProcessingExecution =
                execute(documentProcessingInput, PrepareInputForOcrTask.class)
                        .thenCompose(execute(OcrTask.class));

        invoiceProcessingExecutions.add(singleInvoiceProcessingExecution);
    }
    CompletableFutures.allOf(invoiceProcessingExecutions).get();
} else {
    log.info("No invoices for processing found");
}
```

To send document to OCR process we need to prepare input for OCR Task in a special format with the special key. For that we have special **PrepareInputForOcrTask** task which is in charge of preparing **OcrTaskData** object with "**ocr.task.data**" key in output variables.

The whole example becomes the following:

AP class:

InvoiceProcessingSample.java

```
package eu.ibagroup.samples.ie;

import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModule;
import eu.ibagroup.easyrpa.engine.apflow.TaskInput;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrTask;
import eu.ibagroup.easyrpa.utils.CompletableFutures;
import eu.ibagroup.samples.ie.task.GetIncomingInvoices;
import eu.ibagroup.samples.ie.task.PrepareInputForOcrTask;
import lombok.extern.slf4j.Slf4j;

import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;
```

```
@ApModuleEntry(name = "InvoiceProcessingSample")
@Slf4j
public class InvoiceProcessingSample extends ApModule {

    public TaskOutput run() throws Exception {
        TaskOutput incomingInvoicesResult = execute(getInput(), GetIncomingInvoices.class).get();
        List<String> invoiceDocumentS3PathList = incomingInvoicesResult.get(ExportConstants.
INVOICE_DOCUMENT_S3_PATHS_LIST, List.class);

        List<CompletableFuture<TaskOutput>> invoiceProcessingExecutions = new ArrayList<>();
        if(invoiceDocumentS3PathList.size() > 0) {
            for(String documentS3Path : invoiceDocumentS3PathList) {
                TaskInput documentProcessingInput = new TaskInput(incomingInvoicesResult);
                documentProcessingInput.set(ExportConstants.INVOICE_DOCUMENT_S3_PATH, documentS3Path);

                CompletableFuture<TaskOutput> singleInvoiceProcessingExecution =
                        execute(documentProcessingInput, PrepareInputForOcrTask.class)
                                .thenCompose(execute(OcrTask.class));

                invoiceProcessingExecutions.add(singleInvoiceProcessingExecution);
            }
            CompletableFutures.allOf(invoiceProcessingExecutions).get();
        } else {
            log.info("No invoices for processing found");
        }

        return doNotCareOfResult();
    }
}
```

New class which prepare an input for OCR task:

PrepareInputForOcrTask.java

```
package eu.ibagroup.samples.ie.task;

import eu.ibagroup.easyrpa.engine.annotation.*;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrFormats;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrTask;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrTaskData;
import eu.ibagroup.easyrpa.utils.MlTaskUtils;
import eu.ibagroup.samples.ie.ExportConstants;
import eu.ibagroup.samples.ie.PropertyConstants;
import lombok.extern.slf4j.Slf4j;

import java.util.*;

@ApTaskEntry(name = "Prepare Input For OCR Task")
@Slf4j
@InputToOutput
public class PrepareInputForOcrTask extends ApTask {
    private static final String OCR_TASK_CONFIGURATION_WORK_S3_BUCKET_KEY = "documents_bucket";
    private static final String OCR_TASK_CONFIGURATION_TESSERACT_OPTIONS_KEY = "tesseractOptions";
    private static final String OCR_TASK_CONFIGURATION_IMAGE_MAGICK_OPTIONS_KEY = "imageMagickOptions";
    private static final List<String> OCR_OUTPUT_FORMATS = Arrays.asList(OcrFormats.TEXT, OcrFormats.HOCR,
OcrFormats.JSON, OcrFormats.IMAGE);

    @Input(ExportConstants.INVOICE_DOCUMENT_S3_PATH)
    private String documentS3Path;

    @Output(OcrTask.OCR_TASK_DATA_KEY)
    private OcrTaskData ocrTaskData;

    private String ocrS3WorkBucket;
    private String ocrTesseractOptions;
    private String ocrImageMagickOptions;

    private Map<String, Object> ocrConfiguration = new HashMap<>();
    @AfterInit
    public void init() {
        ocrS3WorkBucket = getConfigurationService().getConfiguration().get(PropertyConstants.
OCR_S3_WORK_BUCKET);
        ocrTesseractOptions = getConfigurationService().getConfiguration().get(PropertyConstants.
OCR_TESSERACT_OPTIONS);
        ocrImageMagickOptions = getConfigurationService().getConfiguration().get(PropertyConstants.
OCR_IMAGE_MAGICK_OPTIONS);

        ocrConfiguration.put(OCR_TASK_CONFIGURATION_WORK_S3_BUCKET_KEY, ocrS3WorkBucket);
        ocrConfiguration.put(OCR_TASK_CONFIGURATION_TESSERACT_OPTIONS_KEY, MlTaskUtils.readObjectFromString
(ocrTesseractOptions, ArrayList.class));
        ocrConfiguration.put(OCR_TASK_CONFIGURATION_IMAGE_MAGICK_OPTIONS_KEY, MlTaskUtils.readObjectFromString
(ocrImageMagickOptions, ArrayList.class));
    }

    @Override
    public void execute() throws Exception {
        log.info("Prepare input for document in '{}' with ocr task with configuration: {}", documentS3Path,
ocrConfiguration);

        ocrTaskData = new OcrTaskData();
        ocrTaskData.setDocumentLocation(documentS3Path);
        ocrTaskData.getFormats().addAll(OCR_OUTPUT_FORMATS);
        ocrTaskData.setConfiguration(ocrConfiguration);
    }
}
```

PropertyConstants.java

```
package eu.ibagroup.samples.ie;

public interface PropertyConstants {
    String S3_BUCKET = "s3.bucket";
    String S3_FOLDER_TO_SCAN = "s3.folder_to_scan";

    String OCR_S3_WORK_FOLDER = "ocr.s3_work_folder";
    String OCR_S3_WORK_BUCKET = "ocr.s3_work_bucket";
    String OCR_TESSERACT_OPTIONS = "ocr.tesseract_options";
    String OCR_IMAGE_MAGICK_OPTIONS = "ocr.image_magick_options";
}
```

ExportConstants.java

```
package eu.ibagroup.samples.ie;

public interface ExportConstants {
    String INVOICE_DOCUMENT_S3_PATHS_LIST = "INVOICE_DOCUMENT_S3_PATHS_LIST";
    String INVOICE_DOCUMENT_S3_PATH = "INVOICE_DOCUMENT_S3_PATH";
}
```

apm_run.properties

```
# EasyRPA Client Configuration File
s3.bucket=ie-sample
s3.folder_to_scan=incoming_invoices

ocr.s3_work_bucket=data
ocr.s3_work_folder=incoming_invoices_work_folder
ocr.tesseract_options=["-l", "eng","--psm","3","--oem","3","--dpi", "800"]
ocr.image_magick_options=["-resample", "450", "-density", "350", "-quality", "100", "-background" , "white" , "-
alpha" , "flatten"]
```

(v. 2.2) Step 3. Apply and execute ML model (IE)

When the model training is completed, you can start using it in your Automation Process. You need to know the model **name** and **versi on**.

EasyRPA provides **MlTask** class to call IE model. You need to prepare input data for MlTask. It should be **MlTaskData** object with "**ml. task.data**" key in output variables.

Code example

To prepare input for MlTask we create **PrepareInputForMlTask** class before we call **MlTask.**

So AP class becomes the following:

InvoiceProcessingSample.java

```
package eu.ibagroup.samples.ie;

import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModule;
import eu.ibagroup.easyrpa.engine.apflow.TaskInput;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import eu.ibagroup.easyrpa.engine.task.ml.MlTask;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrTask;
import eu.ibagroup.easyrpa.utils.CompletableFutures;
import eu.ibagroup.samples.ie.task.GetIncomingInvoices;
import eu.ibagroup.samples.ie.task.PrepareInputForMlTask;
import eu.ibagroup.samples.ie.task.PrepareInputForOcrTask;
import lombok.extern.slf4j.Slf4j;

import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;

@ApModuleEntry(name = "InvoiceProcessingSample")
@Slf4j
public class InvoiceProcessingSample extends ApModule {

    public TaskOutput run() throws Exception {
        TaskOutput incomingInvoicesResult = execute(getInput(), GetIncomingInvoices.class).get();
        List<String> invoiceDocumentS3PathList = incomingInvoicesResult.get(ExportConstants.
INVOICE_DOCUMENT_S3_PATHS_LIST, List.class);

        List<CompletableFuture<TaskOutput>> invoiceProcessingExecutions = new ArrayList<>();
        if(invoiceDocumentS3PathList.size() > 0) {
            for(String documentS3Path : invoiceDocumentS3PathList) {
                TaskInput documentProcessingInput = new TaskInput(incomingInvoicesResult);
                documentProcessingInput.set(ExportConstants.INVOICE_DOCUMENT_S3_PATH, documentS3Path);

                CompletableFuture<TaskOutput> singleInvoiceProcessingExecution =
                        execute(documentProcessingInput, PrepareInputForOcrTask.class)
                                .thenCompose(execute(OcrTask.class))
                                .thenCompose(execute(PrepareInputForMlTask.class))
                                .thenCompose(execute(MlTask.class));

                invoiceProcessingExecutions.add(singleInvoiceProcessingExecution);
            }
            CompletableFutures.allOf(invoiceProcessingExecutions).get();
        } else {
            log.info("No invoices for processing found");
        }

        return doNotCareOfResult();
    }

}
```

Code of **PrepareInputForMlTask:**

PrepareInputForMlTask.java

```
package eu.ibagroup.samples.ie.task;

import eu.ibagroup.easyrpa.engine.annotation.*;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.engine.task.ml.MlTask;
import eu.ibagroup.easyrpa.engine.task.ml.MlTaskData;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrTask;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrTaskData;
import eu.ibagroup.easyrpa.utils.MlTaskUtils;
import eu.ibagroup.samples.ie.ExportConstants;
import eu.ibagroup.samples.ie.PropertyConstants;
import lombok.extern.slf4j.Slf4j;

import java.util.HashMap;
import java.util.Map;
import java.util.UUID;

@ApTaskEntry(name = "Prepare input for ML Task")
@Slf4j
@InputToOutput
public class PrepareInputForMlTask extends ApTask {
    private static final String ML_TASK_CONFIGURATION_WORK_S3_BUCKET_KEY = "documents_bucket";

    @Input(OcrTask.OCR_TASK_DATA_KEY)
    private OcrTaskData ocrTaskOutput;

    @Input(ExportConstants.INVOICE_DOCUMENT_S3_PATH)
    private String documentS3Path;

    @Output(MlTask.ML_TASK_DATA_KEY)
    private MlTaskData mlTaskData;

    private String modelName;
    private String modelVersion;

    private String documentId;

    private Map<String, Object> mlConfiguration = new HashMap<>();

    @AfterInit
    public void init() {
        modelName = getConfigurationService().get(PropertyConstants.IE_MODEL_NAME);
        modelVersion = getConfigurationService().get(PropertyConstants.IE_MODEL_VERSION);

        documentId = UUID.randomUUID().toString();

        String ocrS3WorkBucket = getConfigurationService().getConfiguration().get(PropertyConstants.
OCR_S3_WORK_BUCKET);
        mlConfiguration.put(ML_TASK_CONFIGURATION_WORK_S3_BUCKET_KEY, ocrS3WorkBucket);
    }

    @Override
    public void execute() {
        log.info("Creating ML task input for document located at '{}' with assigned id '{}'", documentS3Path,
documentId);

        mlTaskData = new MlTaskData();
        mlTaskData.setModelName(modelName);
        mlTaskData.setModelVersion(modelVersion);
        mlTaskData.setConfiguration(mlConfiguration);
                MlTaskUtils.prepareIeMlTask(documentId, mlTaskData, ocrTaskOutput);
    }

}
```

PropertyConstants.java

```
package eu.ibagroup.samples.ie;

public interface PropertyConstants {
    String S3_BUCKET = "s3.bucket";
    String S3_FOLDER_TO_SCAN = "s3.folder_to_scan";

    String OCR_S3_WORK_FOLDER = "ocr.s3_work_folder";
    String OCR_S3_WORK_BUCKET = "ocr.s3_work_bucket";
    String OCR_TESSERACT_OPTIONS = "ocr.tesseract_options";
    String OCR_IMAGE_MAGICK_OPTIONS = "ocr.image_magick_options";

    String IE_MODEL_NAME = "ie_model.name";
    String IE_MODEL_VERSION = "ie_model.version";
}
```

ExportConstants.java

```
package eu.ibagroup.samples.ie;

public interface ExportConstants {
    String INVOICE_DOCUMENT_S3_PATHS_LIST = "INVOICE_DOCUMENT_S3_PATHS_LIST";
    String INVOICE_DOCUMENT_S3_PATH = "INVOICE_DOCUMENT_S3_PATH";
}
```

apm_run.properties

```
# EasyRPA Client Configuration File
s3.bucket=ie-sample
s3.folder_to_scan=incoming_invoices

ocr.s3_work_bucket=data
ocr.s3_work_folder=incoming_invoices_work_folder
ocr.tesseract_options=["-l", "eng","--psm","3","--oem","3","--dpi", "800"]
ocr.image_magick_options=["-resample", "450", "-density", "350", "-quality", "100", "-background" , "white" , "-
alpha" , "flatten"]

ie_model.name=ordinary52withfixes
ie_model.version=0.0.2
```

(v. 2.2) Step 4. Read ML output and route documents to Human Task (IE)

To read the ML output data you should get value by "**ml.task.data**" key from **TaskOutput** object. It contains **MlTaskData** object, which contains list of the **Document** objects, so you can iterate through pages inside each document to analyze the extracted entities.

In common case we send one document for processing, so here the example of how to get "**Invoice Number**" extracted field:

```
String extractedInvoiceNumber = null;
MlTaskData mlTaskDataOutput = previousTaskStepOutput.get(MlTask.ML_TASK_DATA_KEY, MlTaskData.class);
Document mlTaskProcessedDocument = mlTaskDataOutput.getDocuments().get(0);
for(Page page : mlTaskProcessedDocument.getPages()) {
        for(Entity entity : page.getEntities()) {
                if(entity.getName().equals("Invoice Number")) {
                        extractedInvoiceNumber = entity.getContent();
                }
        }
}
```

Also usually if some information wasn't extracted by ML, we decide to **send document to Human Task**. You can read more about how to send document to Human task.

Basically to send document to Human Task you need to prepare special input for embedded **HumanTask** class. Input should contains **HumanTaskData** object under "**human.task.data**" key in input.

Code example

In the AP class we add additional function which is asynchronously executed and it reads output from ML step. If "Invoice Number" field is not extracted automatically, we asynchronously execute Human Task. If everything is extracted - we do nothing (we return already completed Completable Future with output from previous step):

InvoiceProcessingSample.java

```
package eu.ibagroup.samples.ie;

import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModule;
import eu.ibagroup.easyrpa.engine.apflow.TaskInput;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import eu.ibagroup.easyrpa.engine.task.ht.HumanTask;
import eu.ibagroup.easyrpa.engine.task.ml.MlTask;
import eu.ibagroup.easyrpa.engine.task.ml.MlTaskData;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrTask;
import eu.ibagroup.easyrpa.utils.CompletableFutures;
import eu.ibagroup.samples.ie.task.*;
import lombok.extern.slf4j.Slf4j;

import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;

@ApModuleEntry(name = "InvoiceProcessingSample")
@Slf4j
public class InvoiceProcessingSample extends ApModule {

    public TaskOutput run() throws Exception {
        TaskOutput incomingInvoicesResult = execute(getInput(), GetIncomingInvoices.class).get();
        List<String> invoiceDocumentS3PathList = incomingInvoicesResult.get(ExportConstants.
INVOICE_DOCUMENT_S3_PATHS_LIST, List.class);

        List<CompletableFuture<TaskOutput>> invoiceProcessingExecutions = new ArrayList<>();
        if(invoiceDocumentS3PathList.size() > 0) {
            for(String documentS3Path : invoiceDocumentS3PathList) {
                TaskInput documentProcessingInput = new TaskInput(incomingInvoicesResult);
                documentProcessingInput.set(ExportConstants.INVOICE_DOCUMENT_S3_PATH, documentS3Path);

                CompletableFuture<TaskOutput> singleInvoiceProcessingExecution =
                        execute(documentProcessingInput, PrepareInputForOcrTask.class)
                                .thenCompose(execute(OcrTask.class))
                                .thenCompose(execute(PrepareInputForMlTask.class))
                                .thenCompose(execute(MlTask.class))
                                .thenCompose(previousTaskStepOutput -> {
                                    MlTaskData mlTaskDataOutput = previousTaskStepOutput.get(MlTask.
ML_TASK_DATA_KEY, MlTaskData.class);
                                                                        Map mlTaskResultMap = MlTaskUtils.
createIeHtOutputMap(mlTaskDataOutput);
                                                                        List<Map<String, Object>>
extractedFields = (List<Map<String, Object>>) mlTaskResultMap.get("entities");

                                                                        boolean invoiceNumberExtracted =
extractedFields.stream().anyMatch(extractedField -> extractedField.get("name").toString().equals
(ExportConstants.IE.INVOICE_NUMBER));
                                                                        if(!invoiceNumberExtracted) {
                                                                                return execute
(previousTaskStepOutput, PrepareInputForHumanTask.class)
                                                                                                .thenCompose
(execute(HumanTask.class));
                                                                        } else {
                                                                                return CompletableFuture.
completedFuture(previousTaskStepOutput); //do nothing
                                                                        }
                                });

                invoiceProcessingExecutions.add(singleInvoiceProcessingExecution);
            }
            CompletableFutures.allOf(invoiceProcessingExecutions).get();
        } else {
            log.info("No invoices for processing found");
        }

        return doNotCareOfResult();
    }
}
```

To send document to Human Task, we add additional step which is responsible for preparing input data for it. Additionally we special boolean variable to output which helps on next steps to identify if record was processed by Human or not.

PrepareInputForHumanTask.java

```
package eu.ibagroup.samples.ie.task;

import eu.ibagroup.easyrpa.engine.annotation.*;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.engine.task.ht.HumanTask;
import eu.ibagroup.easyrpa.engine.task.ht.HumanTaskData;
import eu.ibagroup.easyrpa.engine.task.ml.MlTask;
import eu.ibagroup.easyrpa.engine.task.ml.MlTaskData;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrTask;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrTaskData;
import eu.ibagroup.easyrpa.utils.MlTaskUtils;
import eu.ibagroup.easyrpa.utils.storage.StorageManager;
import eu.ibagroup.samples.ie.ExportConstants;
import eu.ibagroup.samples.ie.PropertyConstants;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.io.FilenameUtils;

import javax.inject.Inject;
import java.util.Map;
import java.util.HashMap;

@ApTaskEntry(name = "Prepare Human Task")
@Slf4j
@InputToOutput
```

```
public class PrepareInputForHumanTask extends ApTask  {

    @Inject
    private StorageManager storageManager;

    @Input(OcrTask.OCR_TASK_DATA_KEY)
    private OcrTaskData ocrTaskOutput;

    @Input(MlTask.ML_TASK_DATA_KEY)
    private MlTaskData mlTaskOutput;

    @Input(ExportConstants.INVOICE_DOCUMENT_S3_PATH)
    private String documentS3Path;

    @Output(HumanTask.HUMAN_TASK_DATA_KEY)
    private HumanTaskData humanTaskData;

    @Output(ExportConstants.DOCUMENT_PROCESSED_BY_HUMAN)
    private boolean documentProcessedByHuman = true;

    private String ocrS3WorkBucket;
    private String documentType;

    @AfterInit
    public void init() {
        ocrS3WorkBucket = getConfigurationService().getConfiguration().get(PropertyConstants.
OCR_S3_WORK_BUCKET);
        documentType = getConfigurationService().getConfiguration().get(PropertyConstants.DOCUMENT_TYPE);
    }

    @Override
    public void execute() throws Exception {
        String documentName = FilenameUtils.getName(documentS3Path);
        log.info("Creating human task for document {} with document type {}", documentName, documentType);

        humanTaskData = new HumanTaskData();
                 humanTaskData.setInputJson(MlTaskUtils.createIeHtInputMap(ocrTaskOutput, storageManager,
ocrS3WorkBucket));
                humanTaskData.setOutputJson(new HashMap(MlTaskUtils.createIeHtOutputMap(mlTaskOutput)));

        humanTaskData.setName(documentName);
        humanTaskData.setDocumentType(documentType);
        humanTaskData.setDescription("Invoice Document: " + documentName);

        log.info("Sending task into workspace {} ", humanTaskData);
    }

}
```

PropertyConstants.java

```
package eu.ibagroup.samples.ie;

public interface PropertyConstants {
    String S3_BUCKET = "s3.bucket";
    String S3_FOLDER_TO_SCAN = "s3.folder_to_scan";

    String OCR_S3_WORK_FOLDER = "ocr.s3_work_folder";
    String OCR_S3_WORK_BUCKET = "ocr.s3_work_bucket";
    String OCR_TESSERACT_OPTIONS = "ocr.tesseract_options";
    String OCR_IMAGE_MAGICK_OPTIONS = "ocr.image_magick_options";

    String IE_MODEL_NAME = "ie_model.name";
    String IE_MODEL_VERSION = "ie_model.version";

    String DOCUMENT_TYPE = "document_type";
}
```

ExportConstants.java

```
package eu.ibagroup.samples.ie;

public interface ExportConstants {
    String INVOICE_DOCUMENT_S3_PATHS_LIST = "INVOICE_DOCUMENT_S3_PATHS_LIST";
    String INVOICE_DOCUMENT_S3_PATH = "INVOICE_DOCUMENT_S3_PATH";
    String DOCUMENT_PROCESSED_BY_HUMAN = "DOCUMENT_PROCESSED_BY_HUMAN";

    interface IE {
        String INVOICE_NUMBER = "INVOICE_NUMBER";
    }
}
```

apm_run.properties

```
# EasyRPA Client Configuration File
s3.bucket=ie-sample
s3.folder_to_scan=incoming_invoices

ocr.s3_work_bucket=data
ocr.s3_work_folder=incoming_invoices_work_folder
ocr.tesseract_options=["-l", "eng","--psm","3","--oem","3","--dpi", "800"]
ocr.image_magick_options=["-resample", "450", "-density", "350", "-quality", "100", "-background" , "white" , "-
alpha" , "flatten"]

ie_model.name=ordinary52withfixes
ie_model.version=0.0.2

document_type=Products invoice
```

(v. 2.2) Step 5. Process extraction result (IE)

After all data is extracted by ML or with help from Human, usually it's reconciled with data from business application using RPA step or inserted inside some application.

In such tasks we need to get extracted data regardless of where it came from ML task or Human task. So in the example bellow you can see how to check where input data came from and simple validation.

Code example

In AP class we added 2 additional steps: **ValidateInvoiceData** which is supposed to be responsible for data reconciliation or data insertion in real world and **PrepareResult** which is supposed to be responsible for sending a report about execution:

InvoiceProcessingSample.java

```
package eu.ibagroup.samples.ie;

import eu.ibagroup.easyrpa.engine.annotation.ApModuleEntry;
import eu.ibagroup.easyrpa.engine.apflow.ApModule;
import eu.ibagroup.easyrpa.engine.apflow.TaskInput;
import eu.ibagroup.easyrpa.engine.apflow.TaskOutput;
import eu.ibagroup.easyrpa.engine.task.ht.HumanTask;
import eu.ibagroup.easyrpa.engine.task.ml.MlTask;
import eu.ibagroup.easyrpa.engine.task.ml.MlTaskData;
import eu.ibagroup.easyrpa.engine.task.ocr.OcrTask;
import eu.ibagroup.easyrpa.utils.CompletableFutures;
import eu.ibagroup.samples.ie.task.*;
import lombok.extern.slf4j.Slf4j;

import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;

@ApModuleEntry(name = "InvoiceProcessingSample")
@Slf4j
public class InvoiceProcessingSample extends ApModule {

    public TaskOutput run() throws Exception {
        TaskOutput incomingInvoicesResult = execute(getInput(), GetIncomingInvoices.class).get();
        List<String> invoiceDocumentS3PathList = incomingInvoicesResult.get(ExportConstants.
INVOICE_DOCUMENT_S3_PATHS_LIST, List.class);

        List<CompletableFuture<TaskOutput>> invoiceProcessingExecutions = new ArrayList<>();
        if(invoiceDocumentS3PathList.size() > 0) {
            for(String documentS3Path : invoiceDocumentS3PathList) {
                TaskInput documentProcessingInput = new TaskInput(incomingInvoicesResult);
                documentProcessingInput.set(ExportConstants.INVOICE_DOCUMENT_S3_PATH, documentS3Path);

                CompletableFuture<TaskOutput> singleInvoiceProcessingExecution =
                        execute(documentProcessingInput, PrepareInputForOcrTask.class)
                                .thenCompose(execute(OcrTask.class))
                                .thenCompose(execute(PrepareInputForMlTask.class))
                                .thenCompose(execute(MlTask.class))
                                .thenCompose(previousTaskStepOutput -> {
                                                                        MlTaskData mlTaskDataOutput =
previousTaskStepOutput.get(MlTask.ML_TASK_DATA_KEY, MlTaskData.class);
                                                                        Map mlTaskResultMap = MlTaskUtils.
createIeHtOutputMap(mlTaskDataOutput);
                                                                        List<Map<String, Object>>
extractedFields = (List<Map<String, Object>>) mlTaskResultMap.get("entities");

                                                                        boolean invoiceNumberExtracted =
extractedFields.stream().anyMatch(extractedField -> extractedField.get("name").toString().equals
(ExportConstants.IE.INVOICE_NUMBER));
                                                                        if(!invoiceNumberExtracted) {
                                                                                return execute
(previousTaskStepOutput, PrepareInputForHumanTask.class)
```

```
                                                                                                .thenCompose
(execute(HumanTask.class));
                                                                        } else {
                                                                                return CompletableFuture.
completedFuture(previousTaskStepOutput); //do nothing
                                                                        }
                                })
                                .thenCompose(execute(ValidateInvoiceData.class))
                                .thenCompose(execute(PrepareResult.class));

                invoiceProcessingExecutions.add(singleInvoiceProcessingExecution);
            }
            CompletableFutures.allOf(invoiceProcessingExecutions).get();
        } else {
            log.info("No invoices for processing found");
        }

        return doNotCareOfResult();
    }
}
```

ValidateInvoiceData just simply check if extracted invoice number field met the length requirement:

ValidateInvoiceData.java

```
package eu.ibagroup.samples.ie.task;

import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.annotation.Input;
import eu.ibagroup.easyrpa.engine.annotation.InputToOutput;
import eu.ibagroup.easyrpa.engine.annotation.Output;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.easyrpa.engine.task.ht.HumanTask;
import eu.ibagroup.easyrpa.engine.task.ht.HumanTaskData;
import eu.ibagroup.easyrpa.engine.task.ml.*;
import eu.ibagroup.samples.ie.ExportConstants;
import lombok.extern.slf4j.Slf4j;

import java.util.List;
import java.util.Map;

@ApTaskEntry(name = "Validate invoice data")
@Slf4j
@InputToOutput
public class ValidateInvoiceData extends ApTask {
         private static final String IE_OUTPUT_ENTITIES_KEY = "entities";
        private static final String IE_OUTPUT_ENTITY_NAME_KEY = "name";
        private static final String IE_TASK_OUTPUT_ENTITY_VALUE_KEY = "content";

    @Input(value = ExportConstants.DOCUMENT_PROCESSED_BY_HUMAN, required = false)
    private boolean documentProcessedByHuman;

    @Input(value = HumanTask.HUMAN_TASK_DATA_KEY, required = false)
    private HumanTaskData humanTaskOutput;

    @Input(value = MlTask.ML_TASK_DATA_KEY, required = false)
    private MlTaskData mlTaskOutput;

    @Output(ExportConstants.INVOICE_VALID)
    private String extractedInvoiceNumber;

    @Output(ExportConstants.INVOICE_VALID)
    private boolean invoiceValid;

    @Override
    public void execute() throws Exception {
        String invoiceNumber;
        if(documentProcessedByHuman) {
            invoiceNumber = extractInvoiceNumberFromHumanOutput(humanTaskOutput);
        } else {
            invoiceNumber = extractInvoiceNumberFromMlOutput(mlTaskOutput);
        }
        invoiceValid = validateInvoiceNumber(invoiceNumber);
        log.info("Validation result is {} for invoice number {}", invoiceValid, invoiceNumber);
    }

         private String extractInvoiceNumberFromHumanOutput(HumanTaskData humanTaskOutput) {
                List<Map<String, Object>> extractedFields = (List<Map<String, Object>>) humanTaskOutput.
getOutputJson().get(IE_OUTPUT_ENTITIES_KEY);
                return extractFieldFromIEOutput(extractedFields, ExportConstants.IE.INVOICE_NUMBER);
        }

        private String extractInvoiceNumberFromMlOutput(MlTaskData mlTaskOutput) {
                Map mlTaskResultMap = MlTaskUtils.createIeHtOutputMap(mlTaskOutput);
                List<Map<String, Object>> extractedFields = (List<Map<String, Object>>) mlTaskResultMap.get
("entities");
                return extractFieldFromIEOutput(extractedFields, ExportConstants.IE.INVOICE_NUMBER);
        }

        private String extractFieldFromIEOutput(List<Map<String, Object>> extractedFields, String
targetFieldName) {
                for(Map<String, Object> extractedField : extractedFields) {
                        if(extractedField.get(IE_OUTPUT_ENTITY_NAME_KEY).toString().equals(targetFieldName)) {
                                return extractedField.get(IE_TASK_OUTPUT_ENTITY_VALUE_KEY).toString();
                        }
                }
                return null;
        }

    private boolean validateInvoiceNumber(String invoiceNumber) {
        return invoiceNumber != null && invoiceNumber.length() > 8;
    }
}
```

PrepareResult simply generates an output with final text, is invoice number valid or not:

PrepareResult.java

```
package eu.ibagroup.samples.ie.task;

import eu.ibagroup.easyrpa.engine.annotation.ApTaskEntry;
import eu.ibagroup.easyrpa.engine.annotation.Input;
import eu.ibagroup.easyrpa.engine.annotation.Output;
import eu.ibagroup.easyrpa.engine.apflow.ApTask;
import eu.ibagroup.samples.ie.ExportConstants;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.io.FilenameUtils;

@ApTaskEntry(name = "Prepare Result")
@Slf4j
public class PrepareResult extends ApTask {

    @Input(ExportConstants.INVOICE_DOCUMENT_S3_PATH)
    private String invoiceS3Path;

    @Input(ExportConstants.EXTRACTED_INVOICE_NUMBER)
    private String extractedInvoiceNumber;

    @Input(ExportConstants.INVOICE_VALID)
    private boolean invoiceValid;

    @Output(ExportConstants.PROCESSING_RESULT)
    private String processingResult;

    @Override
    public void execute() throws Exception {
        String validationResult = invoiceValid ? "valid" : "not valid";
        processingResult = "Invoice document '" + FilenameUtils.getName(invoiceS3Path) + "' with extracted
invoice number '" + extractedInvoiceNumber + "' is " + validationResult;
    }
}
```

PropertyConstants.java

```
package eu.ibagroup.samples.ie;

public interface PropertyConstants {
    String S3_BUCKET = "s3.bucket";
    String S3_FOLDER_TO_SCAN = "s3.folder_to_scan";

    String OCR_S3_WORK_FOLDER = "ocr.s3_work_folder";
    String OCR_S3_WORK_BUCKET = "ocr.s3_work_bucket";
    String OCR_TESSERACT_OPTIONS = "ocr.tesseract_options";
    String OCR_IMAGE_MAGICK_OPTIONS = "ocr.image_magick_options";

    String IE_MODEL_NAME = "ie_model.name";
    String IE_MODEL_VERSION = "ie_model.version";

    String DOCUMENT_TYPE = "document_type";
}
```

ExportConstants.java

```
package eu.ibagroup.samples.ie;

public interface ExportConstants {
    String INVOICE_DOCUMENT_S3_PATHS_LIST = "INVOICE_DOCUMENT_S3_PATHS_LIST";
    String INVOICE_DOCUMENT_S3_PATH = "INVOICE_DOCUMENT_S3_PATH";
    String DOCUMENT_PROCESSED_BY_HUMAN = "DOCUMENT_PROCESSED_BY_HUMAN";
    String INVOICE_VALID = "INVOICE_VALID";
    String EXTRACTED_INVOICE_NUMBER = "EXTRACTED_INVOICE_NUMBER";
    String PROCESSING_RESULT = "PROCESSING_RESULT";

    interface IE {
        String INVOICE_NUMBER = "INVOICE_NUMBER";
        String DEBIT_NOTE_ID = "DebitNoteId";
    }
}
```

apm_run.properties

```
# EasyRPA Client Configuration File
s3.bucket=ie-sample
s3.folder_to_scan=incoming_invoices

ocr.s3_work_bucket=data
ocr.s3_work_folder=incoming_invoices_work_folder
ocr.tesseract_options=["-l", "eng","--psm","3","--oem","3","--dpi", "800"]
ocr.image_magick_options=["-resample", "450", "-density", "350", "-quality", "100", "-background" , "white" , "-
alpha" , "flatten"]

ie_model.name=ordinary52withfixes
ie_model.version=0.0.2

document_type=Products invoice
```