System Requirement

2018-09-18 15:17:55

System Requirement

Supported Platforms

- Ubuntu
- CentOS
- Red Hat Enterprise Linux
- Windows 7
- Windows 10
- Docker

Software Prerequisites

- IBM® Business Process Manager 8.5.x,8.6.0, IBM® Business Automation Workflow V18.X and IBM® Business Automation Workflow V19.0.0.1
- Open Liberty 18.0.0.3 or IBM WebSphere Application Server Version v9.x
  - The appserver JDK is JDK v8+ (64-bit)
- IBM® Db2 v10.x+ or MySQL 5.x+
- Selenium v3.14+

Optional

- A SMTP Server

Browsers Support

- Firefox v52+ (64-bit)
- Chrome v57+ (64-bit)
- IE v11+ (Supports exection only)

Minimal Hardware/VM/Docker Requirements

- 2 core CPU, 4 GB memory and 50 GB disk space.

Recommended Hardware/VM/Docker Requirements

- 2 core CPU, 8 GB memory and 200 GB disk space.

Install IDA Package

Extract the IDA Installation archive file to a specific location. The IDA archive file contains the following directories and artifacts:

- **build** : contains IDA war file that needs to be deployed in Open Liberty.
- **conf** : contains properties file.
- **lib** : contains the required Java library for IDA packaging and setup.
- **sql** : contains the SQL files to create database and tables.
- **toolkit** : contains a required Toolkit TWX file.
- **workspace** : contains the Checkstyle rule projects, which is builded on IBM Operational Decision Manager.
- **package.bat** : Windows command to update IDA war file with client settings.
- **package.sh** : Linux script to update IDA war file with client settings.
- **README.html** : README file.
- **ReleaseNotes.html** : Release notes.

Database Installation and Configuration

Install and Configure DB2

IDA supports DB2 v10.x+. Please refer to the following steps to setup DB2 environment.

Create DB Database

Login into the DB2 server and execute below DB2 commands to create database.

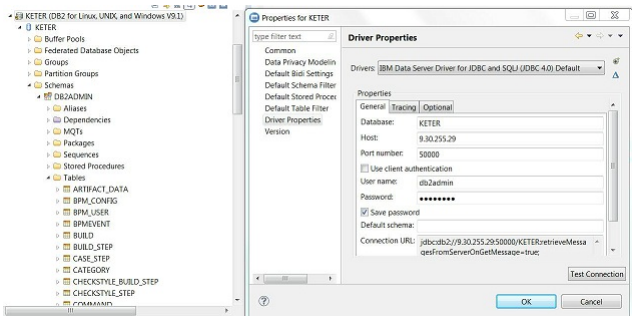
```
db2 create database IDA automatic storage yes using codeset UTF-8 territory US pagesize 32768
db2 connect to IDA
db2 CREATE BUFFERPOOL BP32K IMMEDIATE ALL DBPARTITIONNUMS SIZE AUTOMATIC NUMBLOCKPAGES 0 PAGESIZE 32 K
```

Execute DB Scripts

Start DB2 server and switch to the **IDA** DB and execute DB scripts **schema-db2.sql**, **data-db2.sql** to create the database tables and populate data. These 2 SQL scripts can be found in the **sql** folder of IDA installation package.

```
db2 connect to IDA
db2 -stvf schema-db2.sql
db2 -stvf data-db2.sql
db2 connect reset
```

You can use DB2 client tool (Data Studio) to verify database tables are created and populated with data.

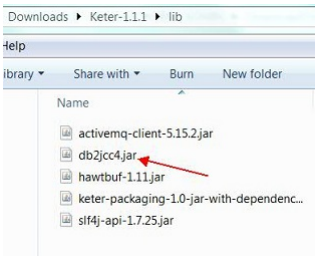


Notes:

- You will want to change [IDA_HOME]/sql/data-db2.sql script so that the organization matches your companies name. Please replace 'IDA' with your company name as outlined below. The default IDA login name and password is "idaAdmin"/"idaAdmin".

Download DB2 Driver
You can download JDBC for DB2 version 11.1

After download, extract the **db2jcc4.jar** from driver package and copy it to the lib folder of IDA installation package. Please note that you can also get the **db2jcc4.jar** from your DB2 server.



Install and Configure MySQL DB

IDA supports MySQL v8.0. Download and install MySQL. Please refer to the following steps to setup MySQL environment.

Create Database

Connect to the MySQL server and use MySQL command to create database.

```
mysql> create database IDA;
```

Execute DB Scripts

Notes:

- You will want to change [IDA_HOME]/sql/data-mysql.sql script so that the organization matches your companies name. Please replace 'IDA' with your company name as outlined below. The default IDA login name and password is "idaAdmin"/"idaAdmin".

Connect to the MySQL server and switch to the **IDA** DB and execute DB scripts **schema-mysql.sql**, **data-mysql.sql** to create the database tables and populate data. These 2 SQL scripts can be found in the **sql** folder of IDA installation package.

You can execute the script with the MySQL source command to execute script. Pls replace **yoursqlpath** to path of the SQL folder location.

```
mysql> use IDA ;
mysql> source yoursqlpath\schema-mysql.sql;
mysql> source yoursqlpath\data-mysql.sql;
```

In MySQL 8.0, the default authentication plugin has changed from **mysql_native_password** to **caching_sha2_password**, and the 'root'@'localhost' administrative account uses caching_sha2_password by default. Please execute below script to use the previous default authentication plugin (**mysql_native_password**) in order to work with IDA.

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'password';
```

Notes:

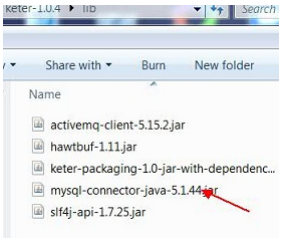
You might need to change **data-mysql.sql** script for the your organization name. Please replace 'IDA' to your company name occur in the below script. The default IDA login name and password is "idaAdmin"/"idaAdmin". Please don't change this username and password in the SQL file.

```
INSERT INTO organization (company_name, active) VALUES ('IDA', true);
INSERT INTO user (user_name, password, role, active, organization_id)
VALUES ('idaAdmin', 'bld2e11b343783a14f6f8362b294bb22', 'ADMIN', true, (SELECT id FROM organization where company_name = 'IDA'));
```

Download mysql Driver
You can download JDBC for mysql version from below link. We recommand you to use mysql-connector-java-5.1.44.jar since it is a tested verson.

mysql-connector-java-5.1.44.jar

After download, copy mysql-connector-java-5.1.44.jar to the lib folder of IDA installation package.



Repacking IDA Application

Properties Configuration

Configure IDA properties

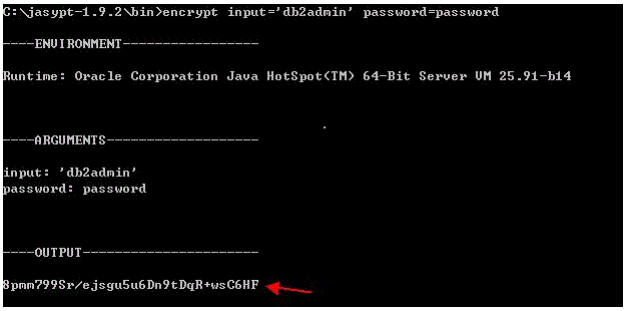
Edit [IDA_HOME]/conf/**ida.properties** under **conf** folder. This properties file contains three main sections.

application configuration

- **spring.datasource.username** : Database connection username
- **spring.datasource.password** : Database connection password
- **spring.datasource.url** : Database connection URL
- **server.host** : IDA server IP or host name
- **server.context-path** : default as "/ida"
- **http.port** : Liberty http port defined in the server.xml
- **https.port** : Liberty https port defined in the server.xml
- **engineConfig.connectionTimeout** : page load timeout
- **engineConfig.waitTimeout** : default wait timeout, used in waitTextPresent and waitElement commands
- **engineConfig.retryTimes** : retry the failed command by the given times
- **engineConfig.retryInterval** : the interval time between the retry.The default is 3 seconds
- **engineConfig.dataDir** : Temp folder to keep resouces like screenshot,checkstyle,twx etc
- **smtp.host** : SMTP host
- **smtp.port** : SMTP port
- **email.from** : Sender Email address
- **spring.activemq.broker-url** : IP and Port of embedded ActiveMQ. It is only needed when you integrate IDA with BPM DEF1 to have the monitoring feature. By default this property is commented out. You can leave it as-is if you don't need the monitoring feature. If you need it, you can uncomment it and replace the localhost with the actual IP address of the IDA deployment server.
- **deployment.timeout** : The timeout in seconds for REST API call to check if snapshot is installed on PS and set as default snapshot
- **deployment.interval** : The interval in seconds for REST API call to check if snapshot is installed on PS and set as default snapshot
- **jasypt.encryptor.password** : default value password
- **enable_case_history** : Enable save test case history function, default value is true.
- **max_case_history** : Set latest test case history max number of each test case, default value is 100.

Notes If you want to use encrypt database password in the **spring.datasource.password**.You need to generate the encrypt password.There are two ways to generate it.The secret key is define in the **jasypt.encryptor.password**. You can use online web tool to generate encrypted password.

Also you can use command line tool from jasypt to generate.



Once you get the encrypt password,you need to change spring.datasource.password like below format.As a general rule, jasypt expects encrypted configuration parameters to appear surrounded by "ENC(…)".

spring.datasource.password: ENC(Yasd1hzrpo05EdNTzlifQ22+8LUNAgcT)

checkstyle properties configuration

- **checkstyle.engine**: The checkstyle engine property can be EMBEDDED or ODM. By default, it's *EMBEDDED*. You need configure the following ODM server and rules configuration if you use ODM as checkstyle engine.
- **resUser** : rule execution server username
- **resPassword** : rule execution server password
- **resUrl** : rule execution server url
- **resPort** : rule execution server port
- **restUrl** : The checkstyle rule ODM REST url

monitoring properties configuration (For BPM Monitoring purpose)

- **event_type** : event types to be stored in IDA database
- **track_serviceflow** : default as true

It is only needed when you want to configure monitoring1 to have the monitoring feature. If you don't need it, you can leave them as-is.

Configure Database properties

You can uncomment DB2 property configuration and comment MySQL property configuration.

```
#DB2 Example
#DB2 Driver
spring.datasource.driver-class-name: com.ibm.db2.jcc.DB2Driver
#DB2 database connection URL
spring.datasource.url: jdbc:db2://9.30.250.7:50000/IDA
```

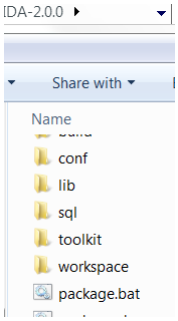
Notes

If you want your IDA support Latin alphabet in page,you need to add **useUnicode=yes&characterEncoding=UTF-8** at the end of jdbc url.

Repacking IDA-web with configuration

The purpose of repackage is to update your application-product.yaml in the war to make sure it loads the recent configuration.Once all the properties in [IDA_HOME]/conf/ida.properties file have been updated, you can then re-package the [IDA_HOME]/build/**ida-web.war** file.

Before running the [IDA_HOME]/**package.bat** or [IDA_HOME]/**package.sh** you will need to set the JAVA_HOME environment variable and make ensure that [JAVA_HOME]/bin is included in the variable. Once the environment variable is set , you can then execute the package shell command to repackage the **ida-web.war**.



Installing IDA Application

Installing IDA Application Components

There are three components for IDA application we need install and configure, included (1) *IDA web application*, (2) *IDA Browser Plugin* and (3) *IDA BPM toolkit*.

Step 1: Installing IDA Web Application

IDA Web Application can be installed on WebSphere Application Server (WAS), liberty or Docker. First, let's introduce the way to install IDA on liberty.

Installing on Liberty

Create a Liberty server manually

You can create a server from the command line.

- Unzip the liberty installation package.Open a command line, then change directory to the wlp/bin directory. Where path_to_liberty is the location you installed Liberty on your operating system.

```
cd path_to_liberty/wlp/bin
```

- Run the following command to create a server. If you do not specify a server name, defaultServer is used. Where server_name is the name you want to give your server.

Windows

```
server create server_name
```

Linux

```
./server create server_name
```

server_name must use only Unicode alphanumeric (for example, 0-9, a-z, A-Z), underscore (_), dash (-), plus (+), and period (.) characters. The name cannot begin with a dash or period. Your file system, operating system, or compressed file directory might impose more restrictions.

If the server is created successfully, you receive message: Server server_name created.

Configure WAS Liberty

- Edit **server.xml** from *wlp/usr/servers/servername* folder. Please ensure both **httpPort** and **httpsPort** are unique and not same with BPM server port.If found port conflict，pls change the **httpPort** and **httpsPort** address.

```
<!-- Enable features -->
   <featureManager>
       <feature>servlet-3.1</feature>
       <feature>ssl-1.0</feature>
       <feature>websocket-1.1</feature>
   </featureManager>

   <!-- This template enables security. To get the full use of all the capabilities, a keystore and user registry are required. -->

   <!-- For the keystore, default keys are generated and stored in a keystore. To provide the keystore password, generate an
        encoded password using bin/securityUtility encode and add it below in the password attribute of the keyStore element.
        Then uncomment the keyStore element. -->
   <!--
   <keyStore password=""/>
   -->
   <webContainer invokeFlushAfterService="false"/>

   <!--For a user registry configuration, configure your user registry. For example, configure a basic user registry using the
       basicRegistry element. Specify your own user name below in the name attribute of the user element. For the password,
       generate an encoded password using bin/securityUtility encode and add it in the password attribute of the user element.
       Then uncomment the user element. -->
   <basicRegistry id="basic" realm="BasicRealm">
       <!-- <user name="yourUserName" password="" />  -->
   </basicRegistry>

   <!-- To access this server from a remote client add a host attribute to the following element, e.g. host="*" -->
   <httpEndpoint id="defaultHttpEndpoint"
                 host="*"
                 httpPort="9081"
                 httpsPort="9443" />


   <!-- Automatically expand WAR files and EAR files -->
   <applicationManager autoExpand="true" startTimeout="360s" stopTimeout="120s"/>
   <application type="war" id="ida" name="ida" location="${server.config.dir}/apps/ida-web.war">
       <classloader delegation="parentLast" />
   </application>

   <keyStore id="defaultKeyStore" password="idaAdmin" />
```

- Copy **IDA-web.war** application into the /usr/servers/*yourservername*/apps directory.

- Start Liberty server and visit the url like http://serverip:port/ida (port is defined in the server.xml).

For example:
In Liberty installation bin folder you can use below command to start the server. **server start default** (default is your server name).

Support http proxy(optional)
We might need proxy server to visit the application,the proxy settings can be passd to the runtime via the JAVA_OPTS environment variable.See below procedure.

- Create a text file named jvm.options.Put it under path_to_liberty/wlp/usr/servers/*yourservername* directory.
- Add following lines based on your acutal proxy setting. You can change https to http as well.
  -Dhttps.proxyHost=host
  -Dhttps.proxyPort=port
  -Dhttps=proxyUser=user
  -Dhttps.proxyPassword=password

with remote testing automation framework based on Selenium Grid.

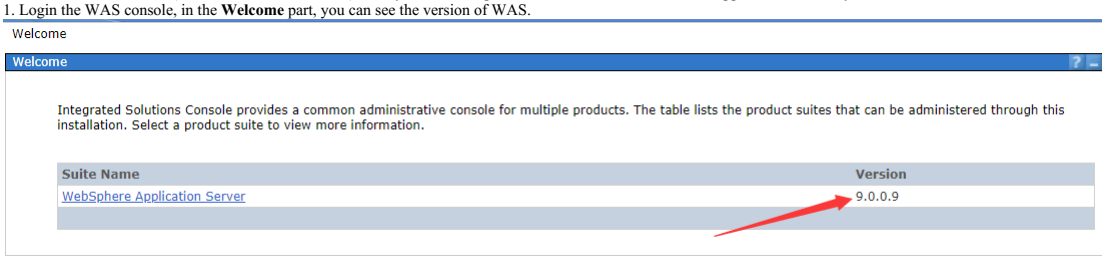
Notes

Below is the reference link for how to setup selenium grid.It includes the detail parameter setting explanation.
- Selenium Grid Setup Guidance

Installing on WAS V9

Check the WAS version

If the WAS version is **9.0** (not 9.0.0.7, 9.0.0.8, 9.0.0.9), it may occur some problems when install the IDA web application, so fisrtly, we should check the WAS version.
1. Login the WAS console, in the **Welcome** part, you can see the version of WAS.



2. If the WAS version is 9.0, you should install the Fix Packs, here are the available fix packs: - 9.0.0.7: WebSphere Application Server traditional V9.0 Fix Pack 7 - 9.0.0.8: WebSphere Application Server traditional V9.0 Fix Pack 8

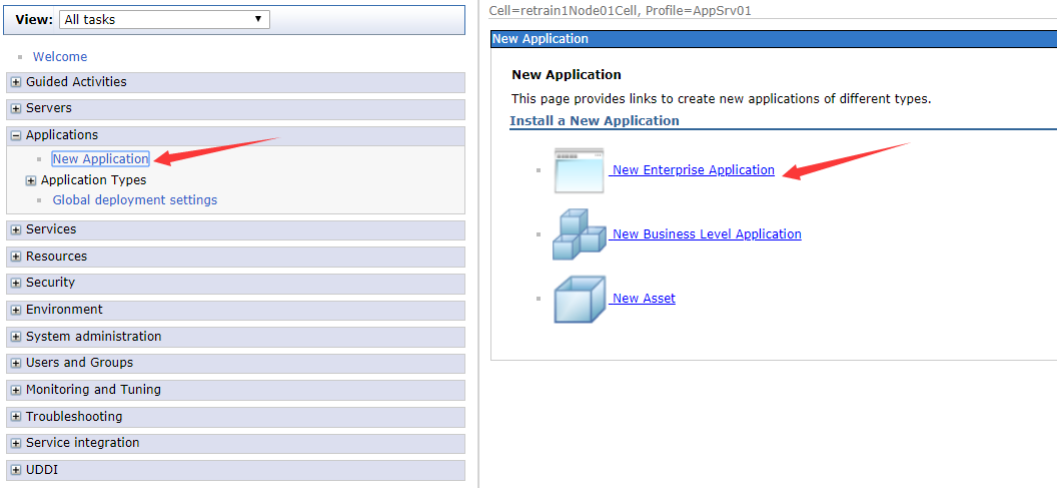
3. When finished the fix packs installation, the version will changed to 9.0.0.7, 9.0.0.8 or 9.0.0.9.

Deploy a New Application on WAS

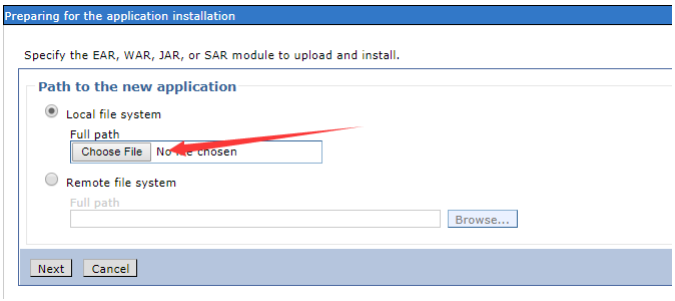
After finishing the installation of the fix packs, the next step is to deploy the IDA war to WAS.

1. Login to the WebSphere Integrated Solutions Console as an administrator (URL: https://host:port/ibm/console/login.do?action=secure).

2. In left navigation bar, click the **New Application>>New Enterprise Application**.

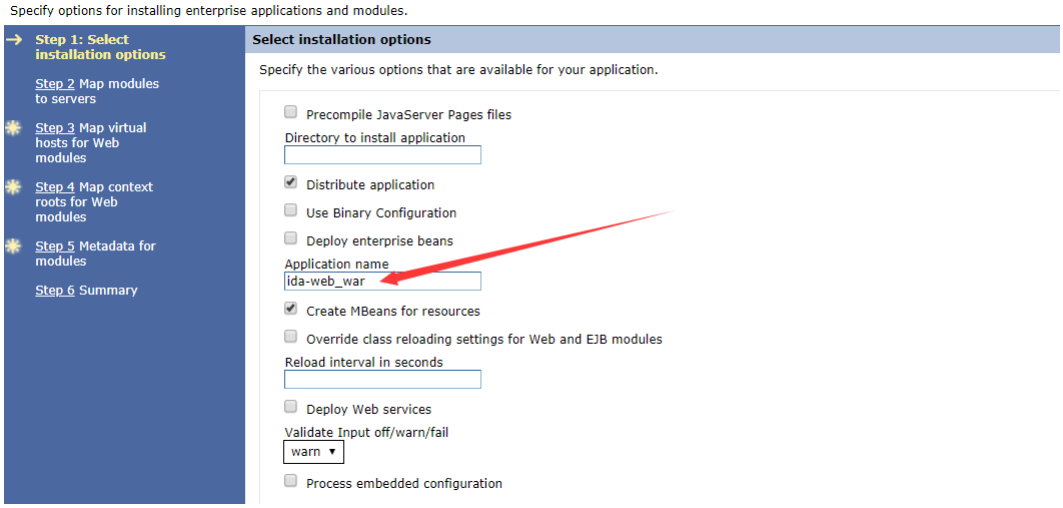


3. In the **Path to the new application** section, check the **Local file system** and select the ida-web.war in your local file system. When the war package is uploaded, click **Next** button.

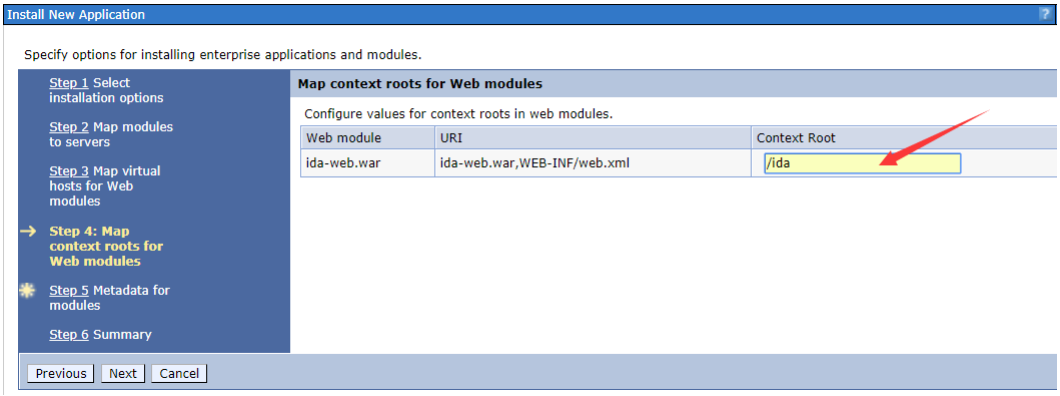


4. Choose the **Fast Path** option. click **Next** button.

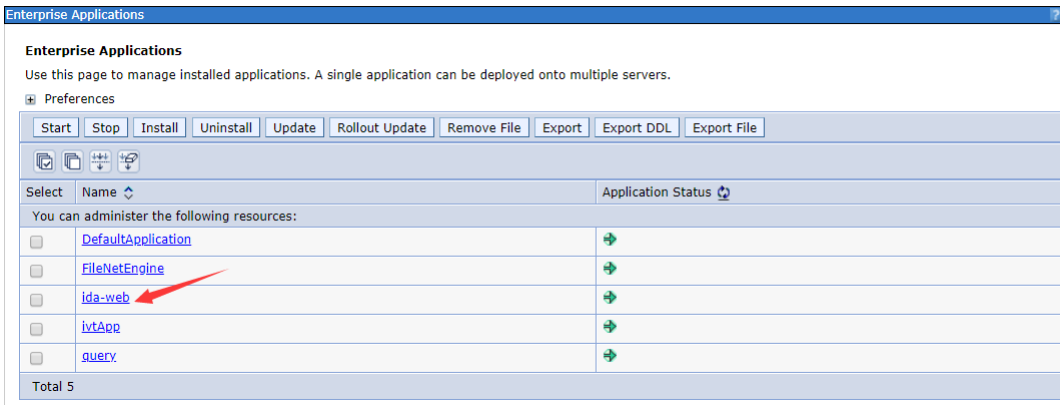
5. Now the current page is used to specify options for installing enterprise application and modules. In step 1, you can change the application name, click **Next** button after changing the application name.



6. There is nothing to change in step 2 and step3. And step 4 is used to configure values for contexts root in web modules, we should set the **Context Root** as **/ida** as shown below.

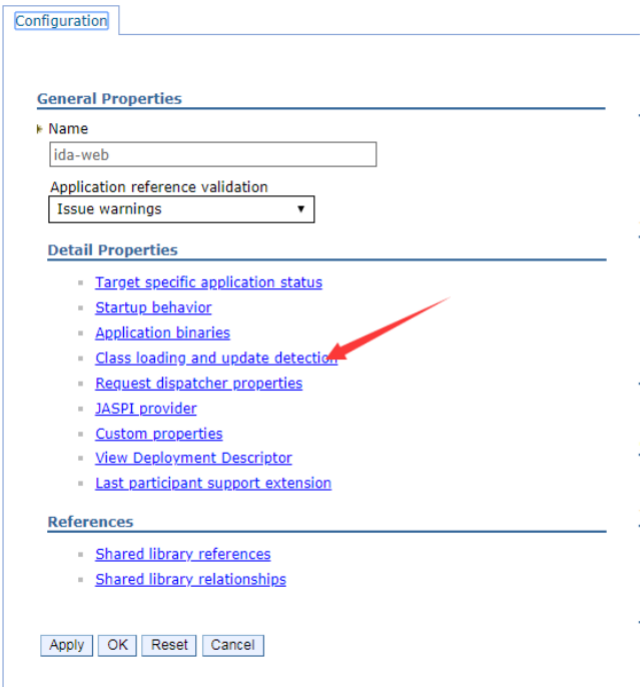


7. There is nothing to change in step 5. In step 6, click **finish** button and wait for the server to complete the installation of IDA web application. When finished, click the **WebSphere enterprise application** in left navigation bar, you can see that the IDA web application is in Enterprise Applications table.

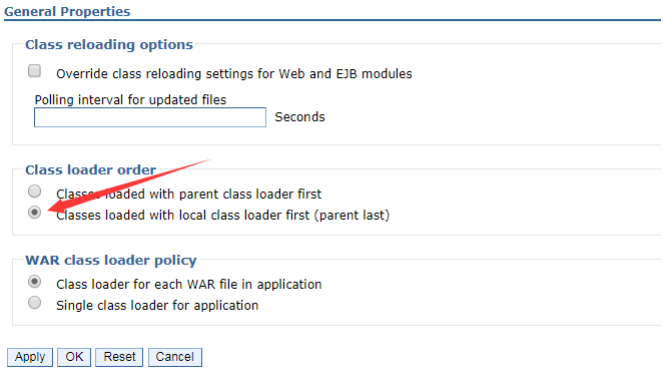


Confige the Class Loader Order

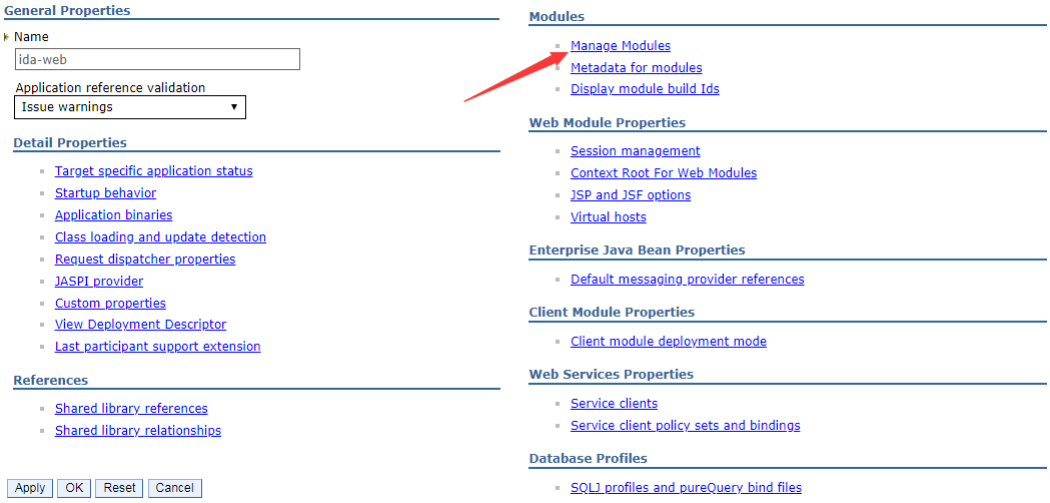
1. Click the link of the **ida-web** in the table and go to the app confiugration page.
2. Click the **Class loading and update detection** link as shown below.



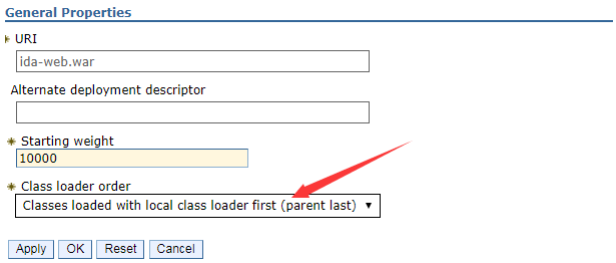
3. Change the class loader order to **Classes loaded with local class loader first (parent last)**.



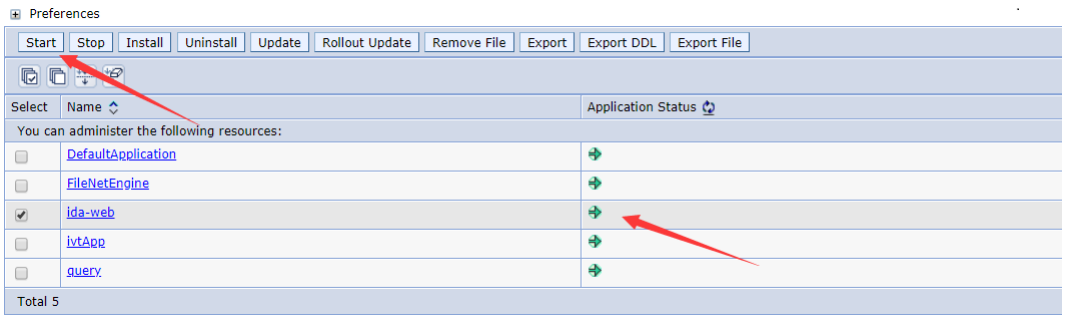
4. Then go back to the configuration page, and then click the **Manage Modules** link.



5. Click the link of **ida-web.war**, in the configuration page, change the class loader order to **Classes loaded with local class loader first (parent last)**.



6. Save the changes and start the IDA web applicaiton. when the status of the IDA web applicaiton changes to **started**, you can visit the url like http://serverip:port/ida to access IDA web application.



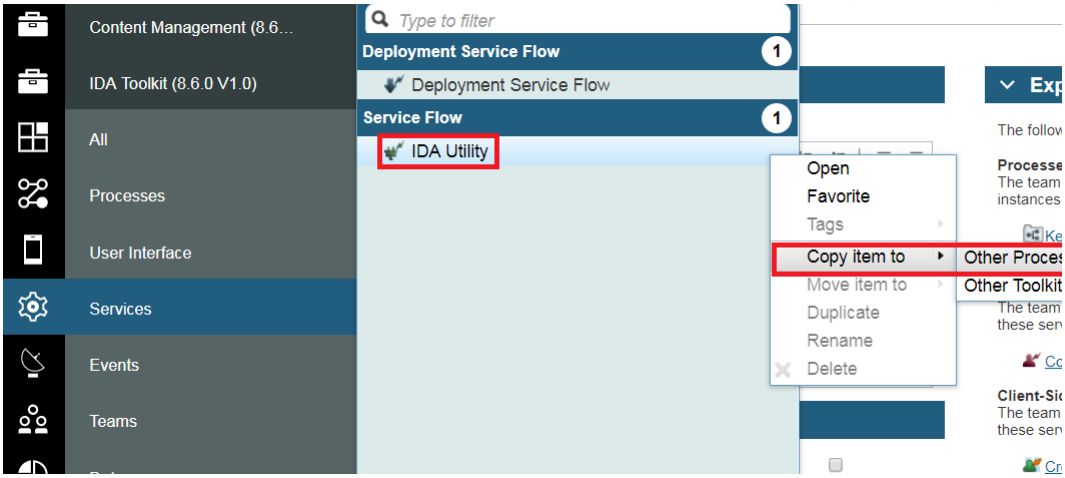
Installing on Docker platform

Refer to IDA-ondocker for deployment steps.

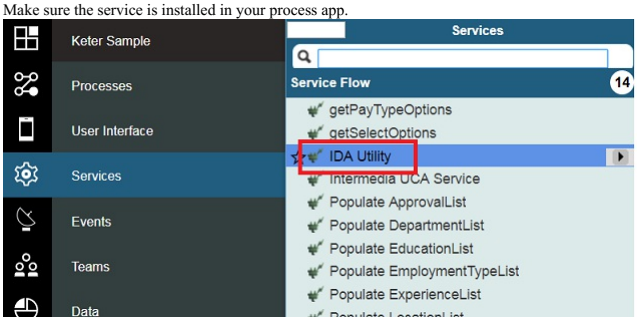
Step 2: Installing IDA BPM Toolkit

The testing capability can only start exposed Business Process, Human Services and AJAX Services. If you wish to directly test other services such as system services, integration services or business processes which are not exposed then you need to install the IDA Toolkit.

1. Import the IDA_Toolkit - 8.6.0_v1.1.twx from IDA toolkit folder into the proces center.
2. Add the IDA toolkit as a this toolkit dependency for within your process application.
3. Right click the “IDA Utility” service flow and copy the item to your process app.



4. Make sure the service is installed in your process app.



Step 3: Installing IDA Browser Plug-in

Chrome

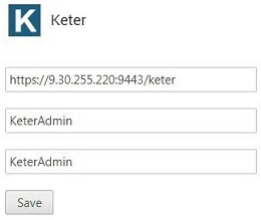
- Open the url https://chrome.google.com/webstore/search/IDA%20IBM
- Click “Add to Chrome” button to install plug-in

Firefox

- Download firefox plugin ida-1.33-fx.xpi
- Drag the “ida-1.33-fx.xpi” file into firefox window
- Click “Add” button

Plug-in Configuration

If you want to use the checkstyle and codereivew feature on web PD, then you need to set the IDA url and user credentials for the plug-in options. the IDA URL: https://9.30.255.220:9443/IDA
the username: the IDA login username the password: the IDA login password.

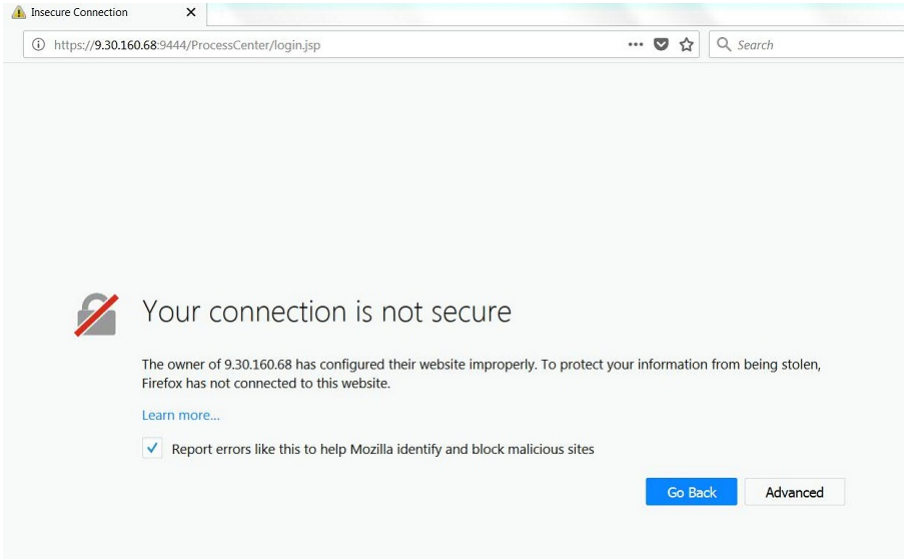


Notes

If you want to install chrome plug-in offline,you can use online https://chrome-extension-downloader.com/ tools,then enter the url https://chrome.google.com/webstore/detail/ida/mjfjiglcnojlicbkomcoohndhpceflbp to download crx ,then install crx.

Self-Signed SSL Certificates Installation

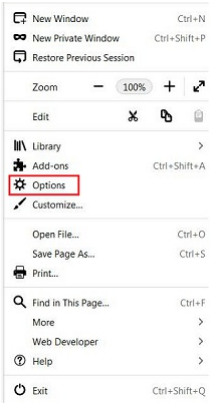
The IDA recorder plugin can't support website with self sign certification by default. In this case, a warning like this:



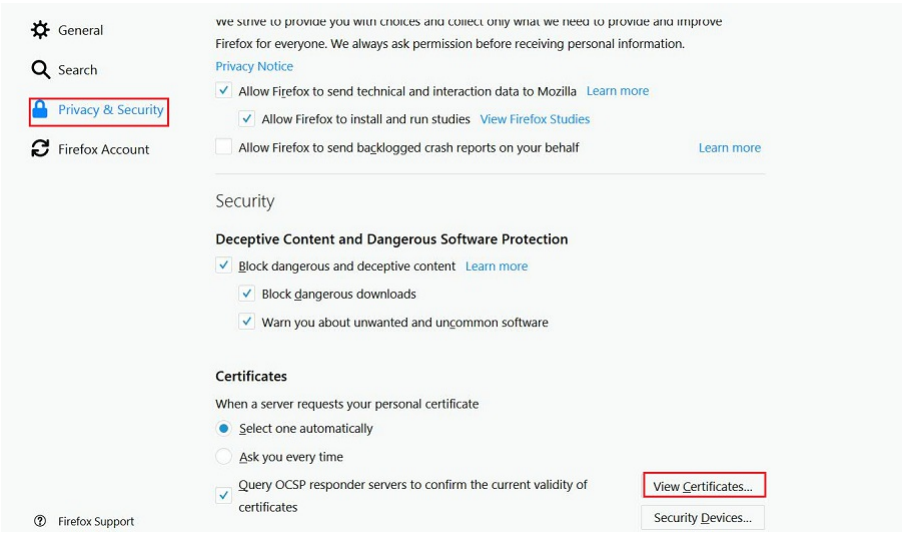
This warning will block the recording of test case. To resolve this problem, we need to make the browsers to accept self-signed certificate.

FireFox - Add a Security Exception

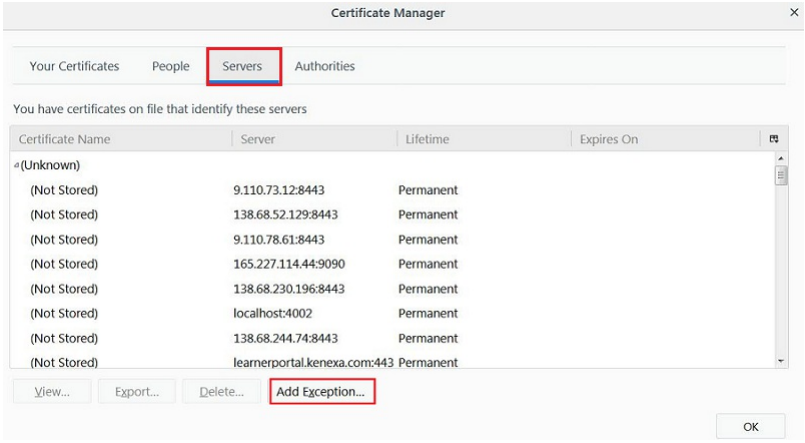
1. In FireFox, go to Tools -> Options.



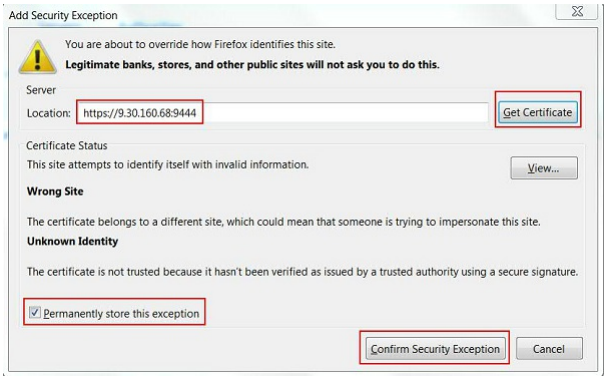
2. Click the **Privaty & Security** tab, then the **View Certificates** button.



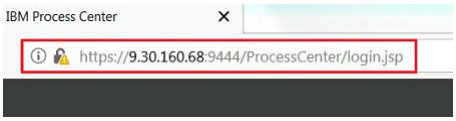
3. Go to the **Services** tab and press the **Add Exception** button.



4. Enter the host and port in **Add Security Exception** dialog, press **Get Certificate** button, check the box near the bottom **Permanently store this exception** and press **Confirm Security Exception** .



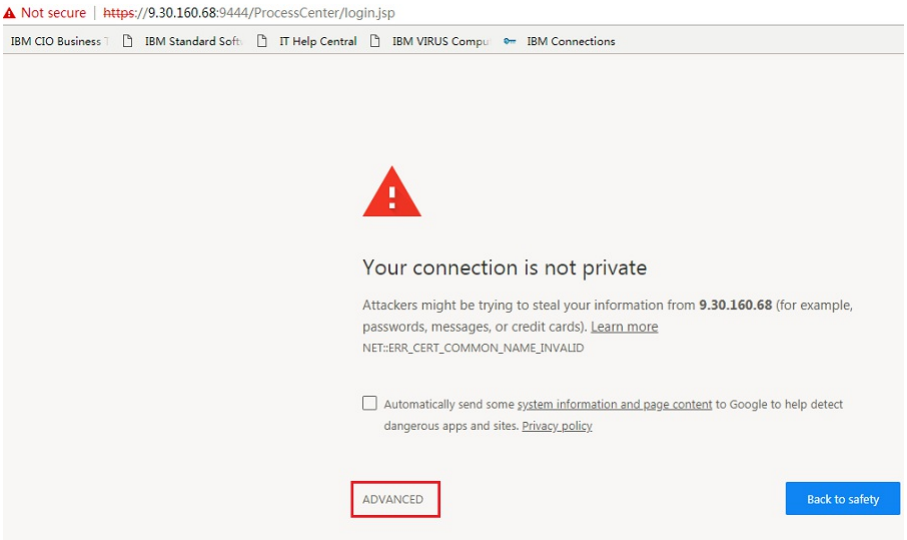
From this point on, FireFox won't show SSL-related errors. when visiting the website, it will like this:



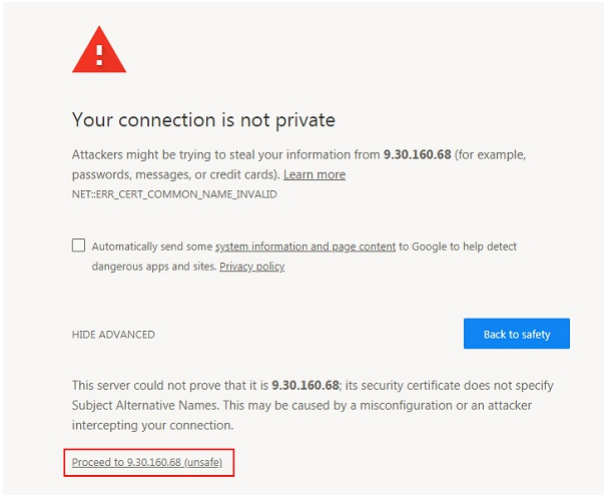
Chrome - Visit in unsafe mode

Chrome browsers can save your data for a short time, and the warning page will not appear and block recording if you visit the testing website in unsafe mode before recording.

1. Click **ADVANCE** in warning page.



2. Click **Proceed to 9.30.160.68(unsafe)**.



Post Installation Setup

Selenium Grid

IDA integrates with remote testing automation framework based on Selenium Grid.

Notes

Below is the reference link for how to setup selenium grid.It includes the detail parameter setting explanation.
- Selenium Grid Setup Guidance
- Internet Explorer Driver Required Configuration

Local installation

1. Download Selenium 3.141.59
2. Download Selenium web drivers and put them under the same folder of the Selenium Jar file.
   - IE Driver
   - Firefox Driver Win64
   - Firefox Driver Linux64
   - Chrome Driver Win32
   - Chrome Driver Linux64
3. Start the Hub
   - Create hubconfig.json `{ "port": 4444, "newSessionWaitTimeout": 10000, "servlets" : [], "withoutServlets": [], "custom": {}, "capabilityMatcher": "org.openqa.grid.internal.utils.DefaultCapabilityMatcher", "registry": "org.openqa.grid.internal.DefaultGridRegistry", "throwOnCapabilityNotPresent": true, "cleanUpCycle": 5000, "role": "hub", "debug": false, "browserTimeout": 60, "timeout": 60 }`
   - Run the command `java -jar selenium-server-standalone-3.141.59.jar -role hub -hubConfig hubconfig.json`
4. Start the Node
   - Create nodeconfig.json `{ "capabilities": [{ "browserName": "firefox", "marionette": true, "maxInstances": 5, "seleniumProtocol": "WebDriver", "version": 67 }, { "browserName": "chrome", "maxInstances": 5, "seleniumProtocol": "WebDriver", "version": 75 }, { "browserName": "internet explorer", "platform": "WINDOWS", "maxInstances": 1, "seleniumProtocol": "WebDriver", "version": 11 }], "proxy": "org.openqa.grid.selenium.proxy.DefaultRemoteProxy", "maxSession": 5, "port": -1, "register": true, "registerCycle": 5000, "hub": "http://localhost:4444", "nodeStatusCheckTimeout": 5000, "nodePolling": 5000, "role": "node", "unregisterIfStillDownAfter": 60000, "downPollingLimit": 2, "debug": false, "servlets" : [], "withoutServlets": [], "custom": {}, "browserTimeout": 60, "timeout": 60 }`
   - Run the command `java -jar selenium-server-standalone-3.141.59.jar -role node -nodeConfig nodeconfig.json`

Docker installation

The followingThe below link is shows a step-by-step introduction to using the official Selenium Docker images using a basic Selemium Hub and Selenium Node configuration.

Selenium: Getting-Started-with-Hub-and-Nodes

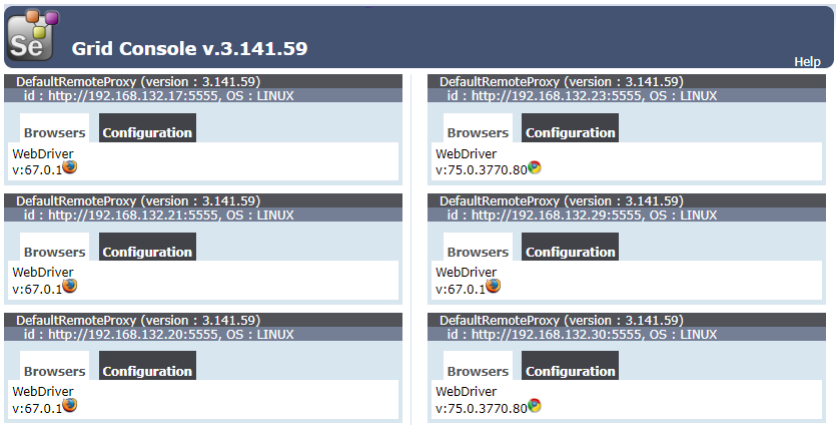
You can follow below script to install Selenium Hub and Node on docker.We recommand you to install debug image version so you can visually see what the browser is doing.

```
docker pull selenium/hub:3.141.59-palladium
docker pull selenium/node-firefox:3.141.59-palladium
docker pull selenium/node-chrome:3.141.59-palladium


docker network create grid
docker run -d -p 4444:4444 --net grid --name selenium-hub selenium/hub:3.141.59-palladium
docker run -d -P --net grid -e HUB_HOST=selenium-hub  -v /dev/shm:/dev/shm selenium/node-chrome:3.141.59-palladium
docker run -d -P --net grid -e HUB_HOST=selenium-hub  -v /dev/shm:/dev/shm selenium/node-firefox:3.141.59-palladium
```

You can do the below command mutiple times,it will create mutiple chrome/firefox node for the grid remote testing. It can provide you to run the selenium tesing parallel in the Grid.We recommand you to create above 5 node number for chrome/firefox each.

```
docker run -d -P --net grid -e HUB_HOST=selenium-hub  -v /dev/shm:/dev/shm selenium/node-chrome:3.141.59-palladium
docker run -d -P --net grid -e HUB_HOST=selenium-hub  -v /dev/shm:/dev/shm selenium/node-firefox:3.141.59-palladium
```

After that you can open the browser http://dockerhostIP:4444/grid/console to check selenium grid/node installation status.



Monitoring configuration (Optional)

You could refer to Monitoring configuration to enable monitoring feature of IDA.

Task 1: Create a test project and generate test cases

IDA enables testers to automate their test scenarios easily and provides 60+ built-in automation test commands to support BPM application test.

Pre-requisites

Before you start the tutorial, you need to setup below configurations with the administrative account ("idaAdmin"/"idaAdmin").

Users management
You need to setup user account for the IDA login credentials.

BPM configuration
You need to setup BPM server and make the BPM account be ready.

Selenium hub configuration
You need to setup selenium node url for the remote testing.

About this task

In this task, you will create a new test project after you create test cases based on this test project.

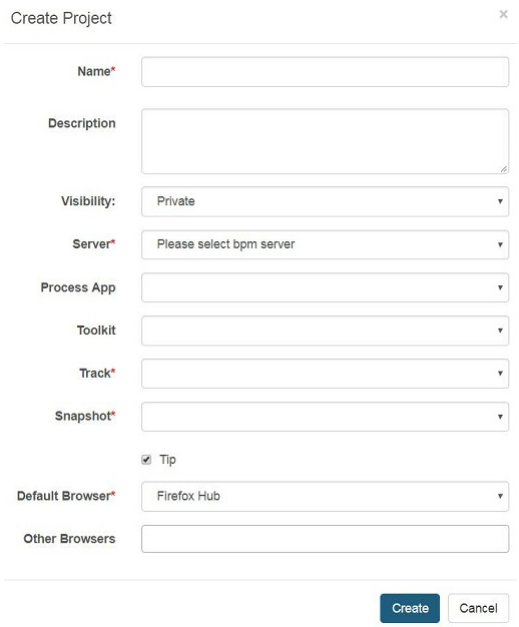
Step 1: Create a test project

In this step, you will create an empty project.

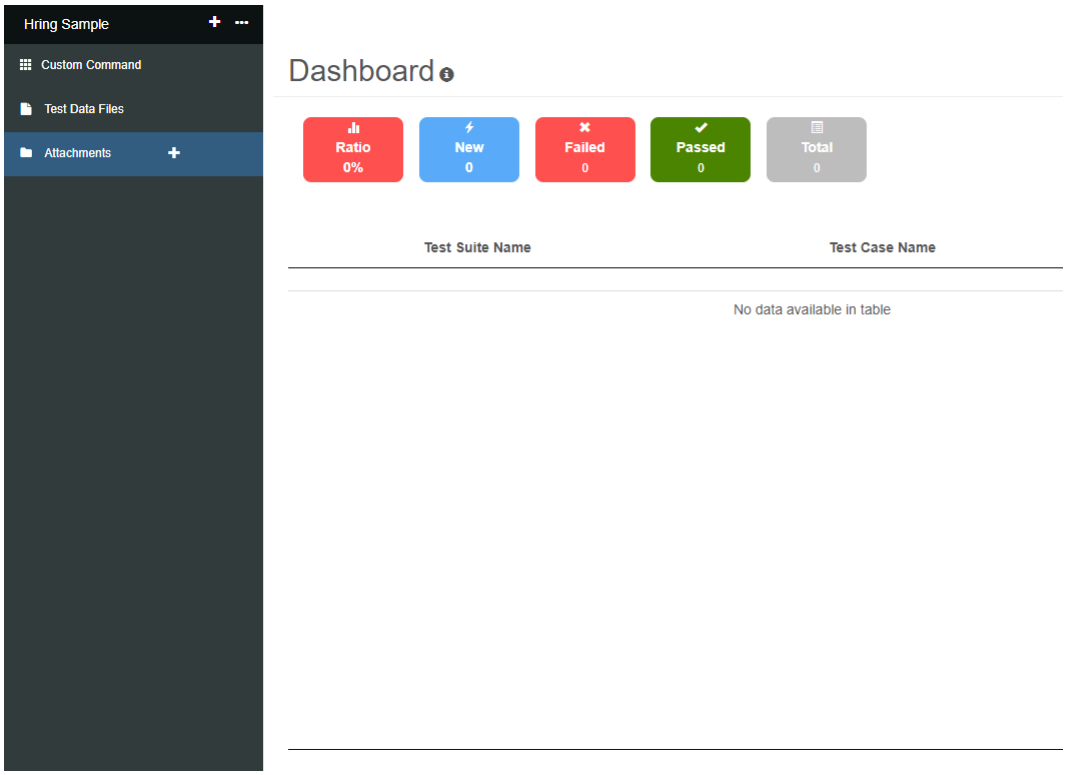
Procedure

To create a new test project:

1. In this project list page, click **Create Project** button. Name the project name as **Hiring Sample**. Choose the values like below screenshot. For the visibility, you can choose private or team you belong to.



2. Click **Create** button, you will enter into project dashboard page.



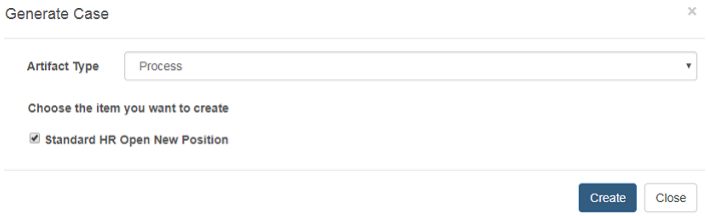
Step 2: Generate test cases

In this step, you'll generate test cases based on new created test project. You can create cases based on the **Artifact** Type.

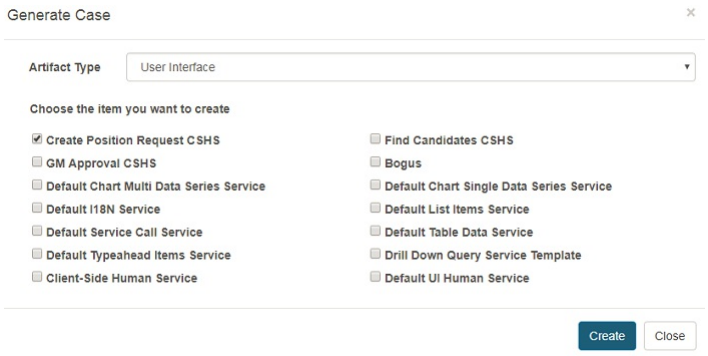
Procedure

To generate a process test case and an user interface test case:

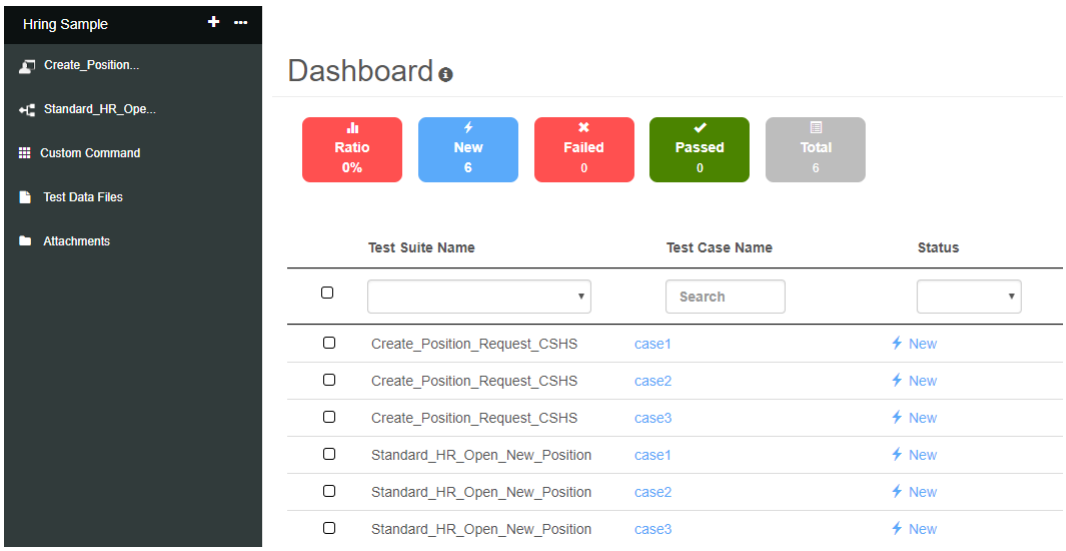
1. In the left-side menu bar, click + button on the right of project name.
2. Choose **Process** in the **Artifact Type** field and **Standard HR Open New Position** in the **Items** field.



3. Click **Create** button.
4. Click + button on the right of project name.
5. Choose **User Interface** in the **Artifact Type** field and **Create Position Request CSHS** in the **Items** field.



6. Click **Create** button.You can see two test suites with related test cases are generated in the project dashboard.



Task 2: Record, replay and run a test case

IDA provides a browser plugin recorder, which allows users to record their test scenarios without coding.

About this task

In this task, you can record, replay and run a test case that you generated in the previous task.

Step 1: Set up assignees for case steps

In this step, you need to assign the BPM task users for case steps. Take the case1 of **Create Position Request CSHS** as an example.

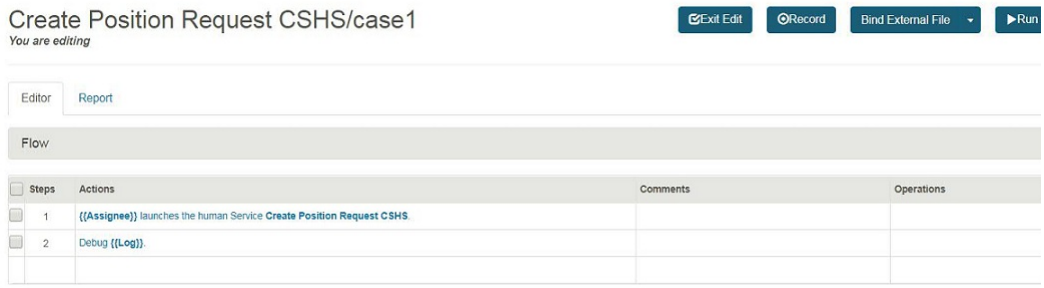
Procedure

To set up assignees for case steps:

1. Click **Create Position Request CSHS** in left-side menu bar.
2. Click **case1**.
3. Click **Edit** button on the right of suite name.

4. Click button of case step which includes “Assignee” in **actions** column.

5. Select **administrator** in **Assignee** field in order to specify other BPM users to help you complete the business process.



6. Click **Exit Edit** button to finish edit.

Step 2: Record a test case

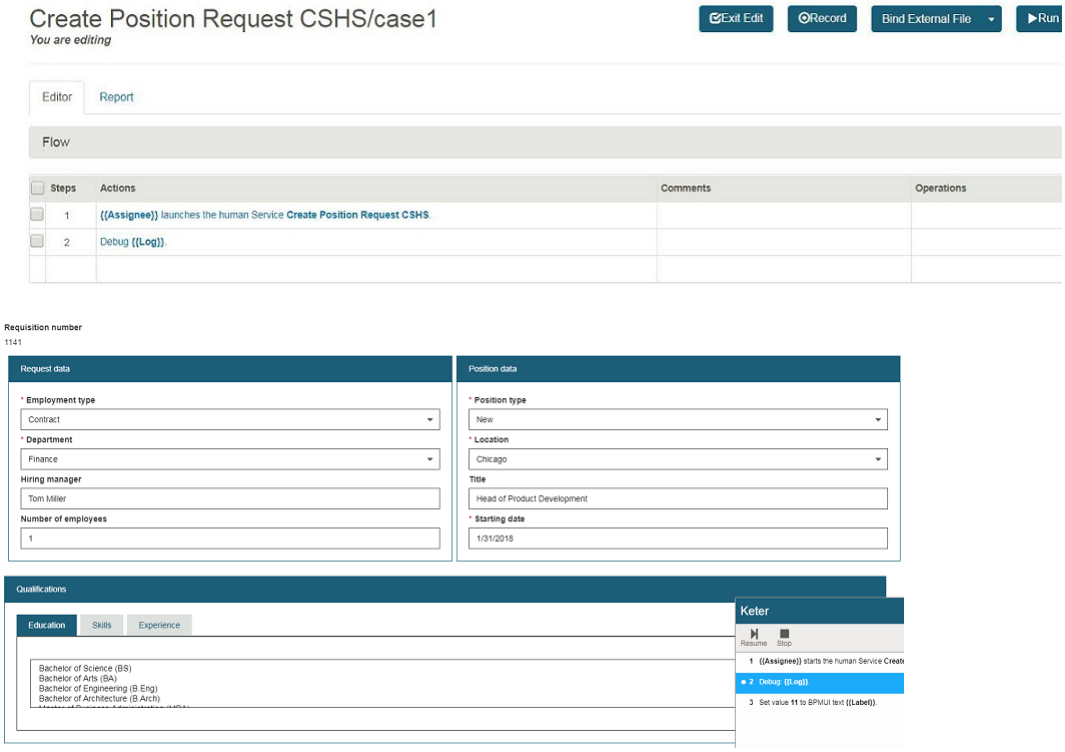
In this step, you can record a test case according to your inputs on the web UI.

Procedure

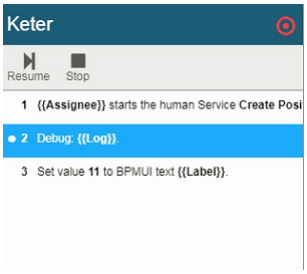
To record a test case:

1. Click **Edit** button on the right top of page.

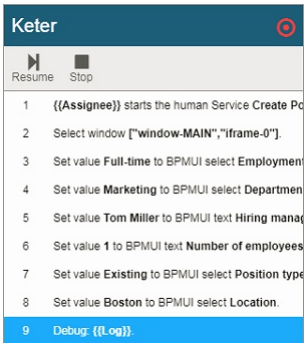
2. Click **Record** button on the right top of page. It will the launch UI record page.



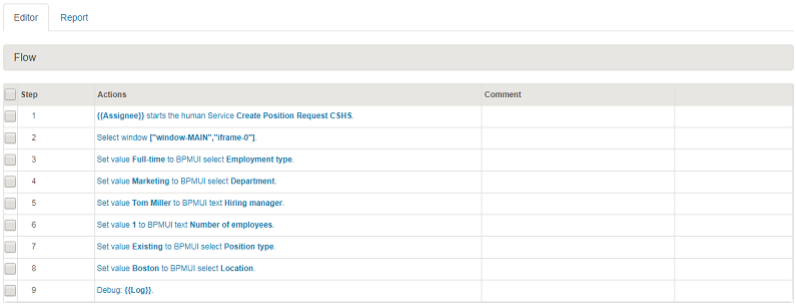
3. In the **Debug {{log}}** step, the process will be stopped. You need to click button to resume.



4. Record every UI action, you can see each step you have done in tool bar.



5. Click button to finish the recording. You can see the recorded case step in the list table.



Step 3: Replay a test case

In this step, you can replay a test case that you record in the previous step.

Procedure

To replay a test case:

1. Click **Record** button on the right of case name to replay it.
2. Click ■ button to finish the replay.
3. Click **Exit Edit** button on the right of suite name.

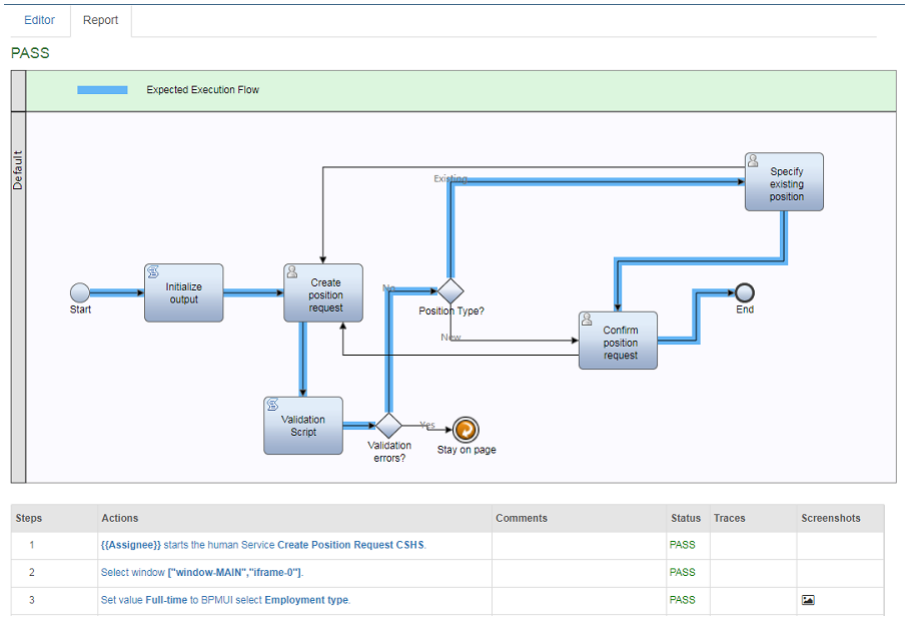
Step 4: Run a test case

In this step, you can run a test case.The running of automated testing is asynchronous.

Procedure

To run a test case:

1. Click **Run** button on the right of case name.
2. Click **report** button to see detail report including process execution flow.
3. Check every case step result status with “PASSED” or “FAILED” in case step list table.
4. Click the **Screenshots** icon link to see the captured screenshot.



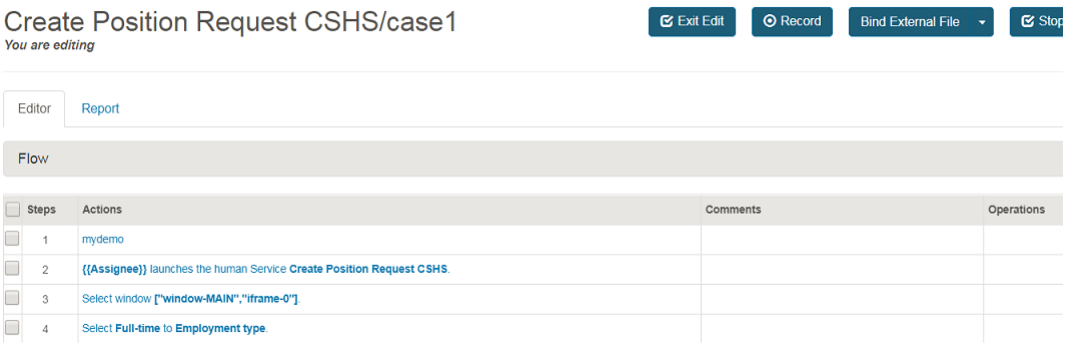
Steps	Actions	Comments	Status	Traces	Screenshots
1	{{Assignee}} starts the human Service Create Position Request CSHS.		PASS		
2	Select window ["window-MAIN","iframe-0"].		PASS		
3	Set value Full-time to BPMUI select Employment type.		PASS		

Step 5: Stop a test case

In this step, you can stop a test case.

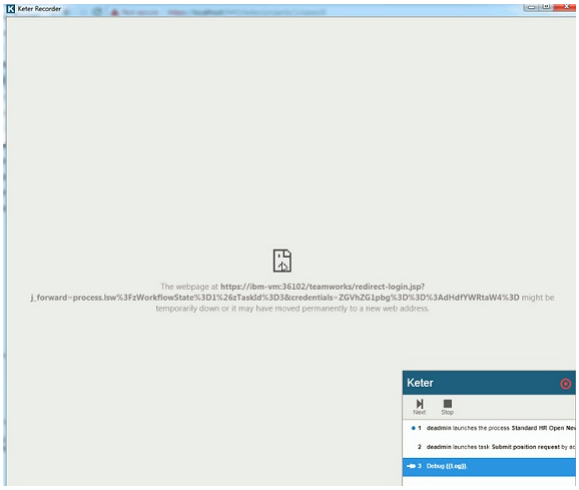
Procedure

Click **Stop** button on the right of case name.



Notes

When you fist record and play your test case you might see the below error.



This is because IBM Business Process Manager uses default certificates that are not issued by a trusted certificate authority (CA). The untrusted certificate can cause errors in the user's browser and possibly block the functions of the OpenPages' integration, including the Process Portal home page.

In order to slove this issue,you need to copy the url from above screenshot and launch this url from the browser and accept the ssl security warning,then try to record/replay again.

Task 3: Set up Checkstyle

IDA enables developers to perform static analysis on the BPM application, which enforces design convention and processes development standard. There are 80+ checkstyle rules are defined based on BPM development best practices and anti-patterns. The rules have been implemented with POJO engine. Refer to Customize Checkstyle rules about rules custmization.

About this task

In this task,you can set up Checkstyle of the project that you created in the previous task.

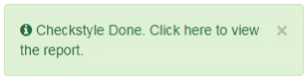
Procedure

Set up Checkstyle of a project:

1. Click button for **Hiring Sample** in project list page and the report will be shown in a few seconds. The analyzing of checkstyle is asynchronous that you could do other things when it is performing the checkstyle.

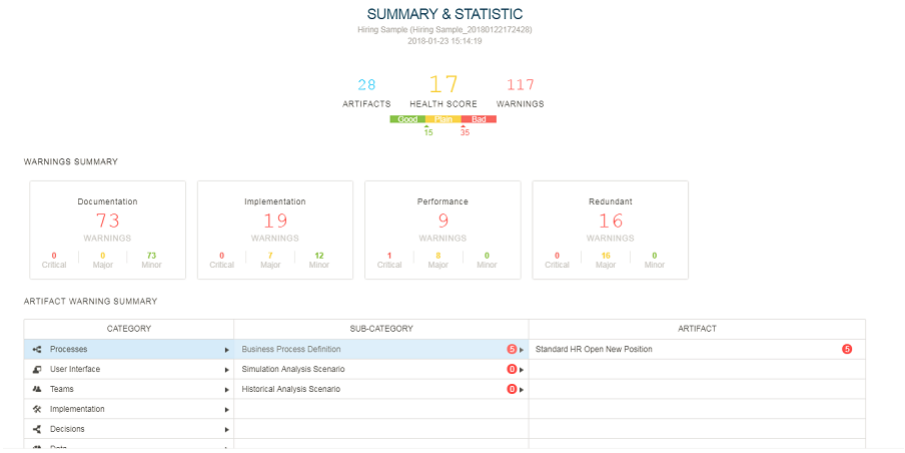
   

   Once completed, the notification will be appeared in the lower right corner of the screen.

   You can also analyze any of your BPM projects in the **Checkstyle** tab. More Detailed.

2. When the analysis is over, the analysis report will be displayed as below. Details of the report.



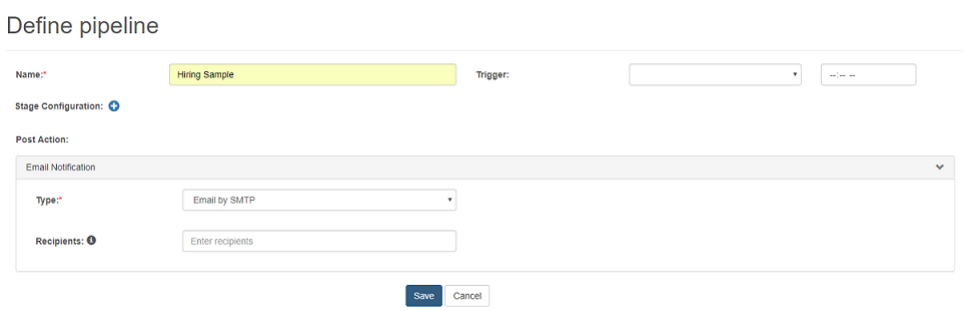
Task 4: Create a pipeline for continuous deployment

IDA provides a build-in continuous integration and deployment pipeline to orchestrate testing, checkstyle, snapshot taking and deployment. ### About this task Before you start,make sure you have set up the test project. ### Step 1: Create a pipeline

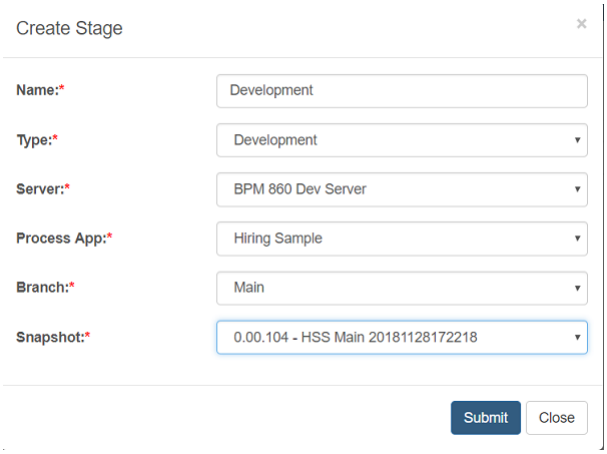
We provide a web UI for users to define the pipeline easily.

Procedure

1. Click **Create Pipeline** button in pipeline page. Fill the values like below. Check the **unit test**, **Snapshot**, **Checkstyle** for the dev env configuration.



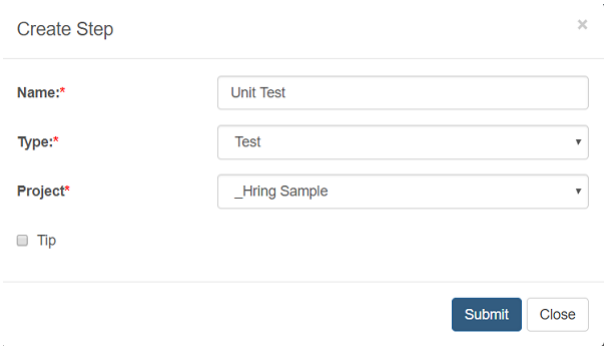
2. Click button. Then fill in **Create Stage** form like the screenshot and **Submit**. Remember to select the first option under snapshot column as it's the latest snapshot.



More details of the parameters in the table, pls see **Create a pipeline for continuous deployment**.

3. Click button shown below.

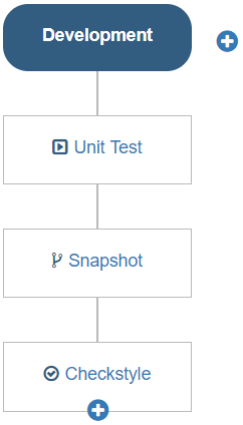
Fill in the form like below and **Submit**.



4. Repeat the step 3 to add **Snapshot** and **Checkstyle** step and fill out the forms according to the table below. The button clicked is the one below the stage step.

	Snapshot	**Checkstyle**
Name	Snapshot	Checkstyle
Type	Snapshot	Checkstyle
Naming Pattern	{MAJOR_MINOR.PATCH}-{APP_ACRONYM}{TRACK}{yyyyMMddHHmmss}	N/A
Health Score Threshold	N/A	20
Warnings Threshold	N/A	200
Ignore Empty Documentation	N/A	Check

Finally, the stage will be like this.



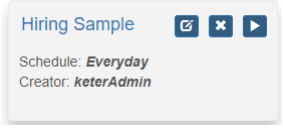
5. Click **Save** button.

Step 2: Run the pipeline

Now you can run the pipeline.

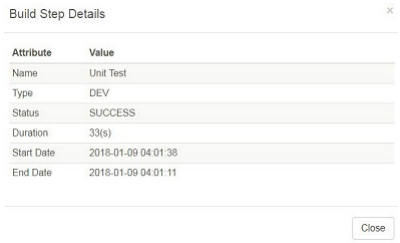
Procedure

1. In the pipeline page, click **Run** icon at the top right corner of the pipeline section.



When you run the build, the pipeline build view will appear with every stage's building status. Blue color means running, grey means waiting, green means success and red means failure.

When you click on **enlargement** icon of build step, you will see the build step details. It will include build time, duration etc information.



2. When the pipeline build is over, click the **build no.** link to see the whole report. Details of the report .

Summary

In this tutorial you know how to use IDA for a working CI/CD solution for BPM process applications.

The tutorials have shown the following modules to you:

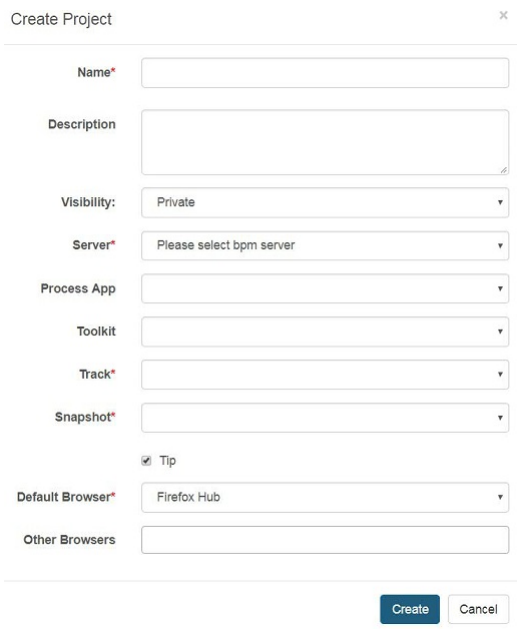
- **Test**: An web-based UI editor for users to record BPM UI and test BPM application.
- **CheckStyle**: Static code analysis for BPM application.
- **Pipeline**: A build-in continuous integration pipeline for testing, checkstyle, snapshot , artifact deployment(online and offline deployment.

For more you could explore our demos on YouTube.

Test project management

Create a test project

1. In the project list page, click **Create Project** button.



2. Fill in the project form according to the following table.

Parameter	Description
Project Name	Name of project
Project Description	Description of project
Server	Server environment
Process App	Process apps of the selected server
Toolkit	Toolkits of selected server
Branch	The branch of selected process app or toolkit
Snapshot	The snapshot of selected branch

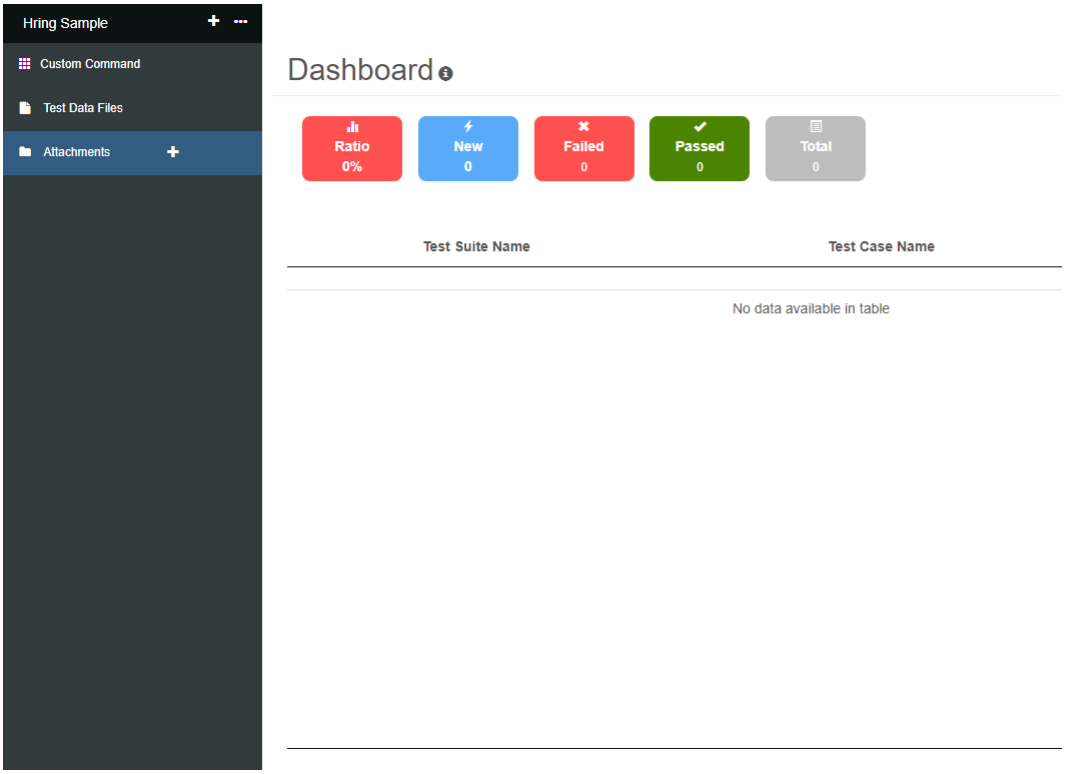
Parameter	Description
Selenium Hub	Support chrome, firefox selenium node
Tip	Using the latest snapshot
Migrate an old project	Support test case project migration from eclipse version

Notes:

Remember to select the process app or toolkit, which can't be both selected. You need to upload Eclipse test project zip file when you check the "Migrate an old project".

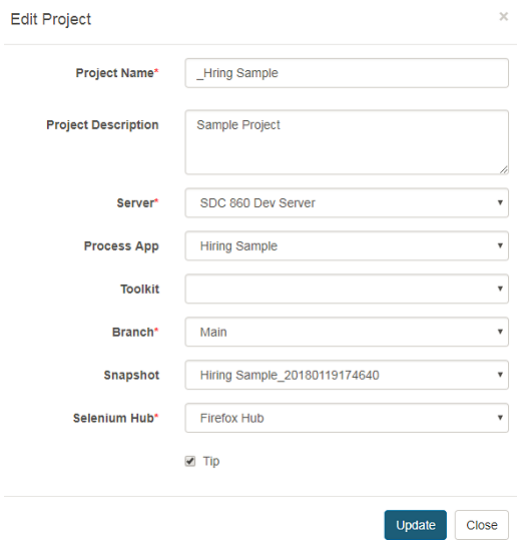
Known Limitions:
The test project migration is only supported under development environment.
It doesn't support the java/js customization test project.

3. Click **Create** button, you will enter into project dashboard page.



Edit a test project

1. Click button on the project list page.



2. Update project form according to the table showed in the creation session.

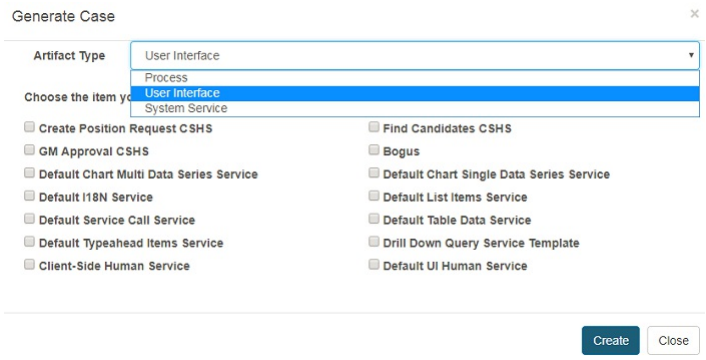
Delete a test project

Click ✖ button on the project list page.

Test case management

Generate test cases

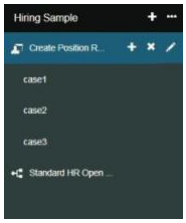
1. In the left-side menu bar, click + button on the right of project name.
2. Click **Artifact Type** field, you can see three kinds of artifacts: Process, User Interface and System Service. Choose **User Interface** or **System Service** to generate a unit test case and choose **Process** to generate a BPD test case.



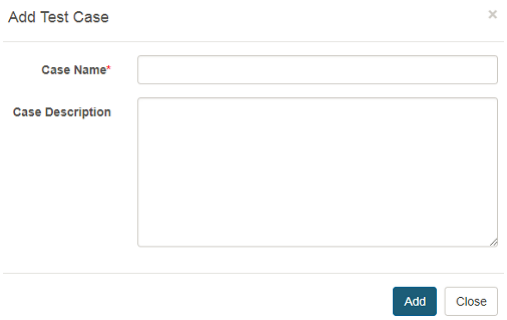
3. Choose process items.
4. Click **Create** button to generate cases.

Create a test case

1. Click + button on the right of suite name.



2. Fill case name and case description.

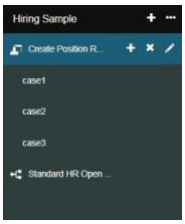


3. Click **Add** button. You can see new test case shown in left-side menu bar. When your mouse hover on button, you can see the description of the test case.

Edit test case

Edit test suite info

1. Click button on the right of suite name.

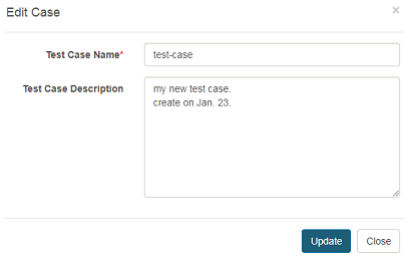


2. Update suite name.
3. Click **Update** button.

Edit test case info

1. Click button on the right of case name.

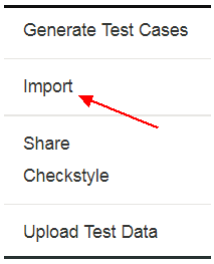
2. Edit test case name and description.



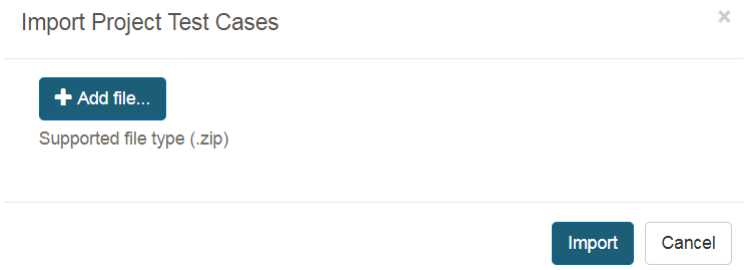
3. Click **Update** button.

Import test case

1. Click **…** link on the right of suite name and popup the menu.
2. Click "Import" link on the popup menu.

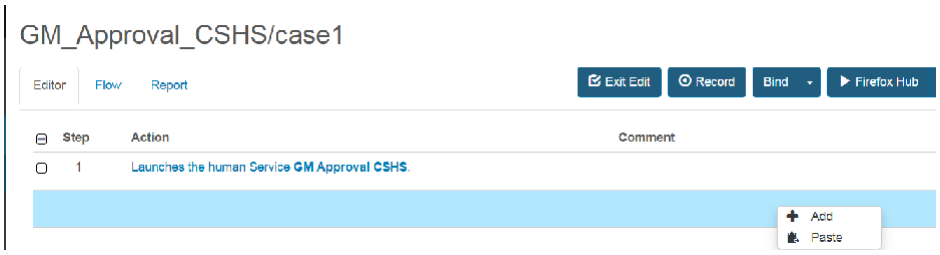


3. Click the "Add file" button to upload the zip file then click the "Import" button to migrate test cases.

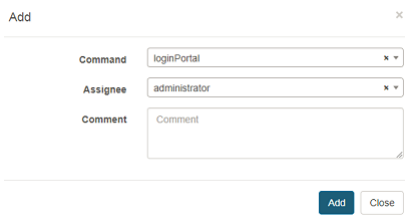


Add a case step

1. Select the test case step position you want to insert.In the popup menu choose **Add** link to insert.



2. Choose command you want to add and add comment for command if necessary. Different commands have different parameters. More details see API reference. Fill form to add case step.



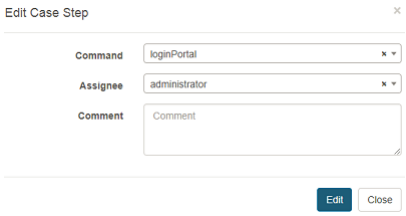
3. Click **Add** button.

Edit a case step

1. Select the case step you want to edit.

2. In the popup menu click the edit link.

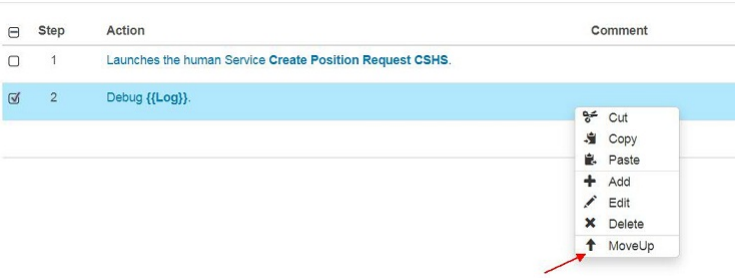
3. Update case step.



4. Click **Edit** button.
5. Click **Exit Edit** button on the right of suite name.

Adjust the order of case step.

1. Select the case step you want to move.
2. In the popup menu, click **MoveUP** or **Movedown** link to adjust the order of a case step.



3. click **Exit Edit** button on the right of suite name.

Copy case steps

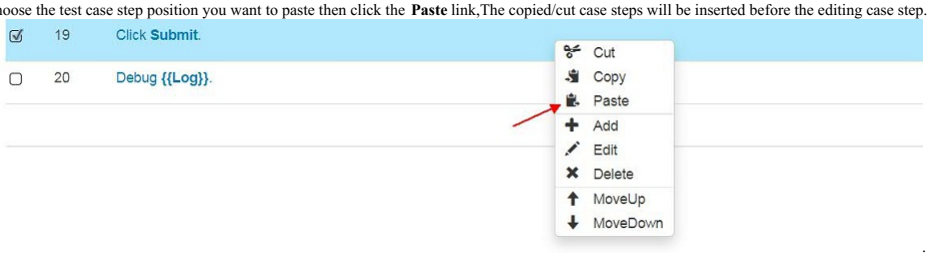
1. Click **Edit** button on the right of suite name.
2. Select the case steps to be copied,in the popup menu click **Copy** link.

Cut case steps

1. Click **Edit** button on the right of suite name.
2. Select the case steps to be cut, in the popup menu click **Cut** link.

Paste case steps

1. Click **Edit** button on the right of suite name.
2. Choose the test case step position you want to paste then click the **Paste** link,The copied/cut case steps will be inserted before the editing case step.



Delete a case step

Select the test case steps in the test case step table.In the popup menu click **delete** link.

Delete a test case

Click ✖ button in test case list from left-side menu.

Test Case data binding

Create a csv test data file

1. Open the excel and edit data like below format. The first column is the key name of data.

	A	B	C	D	E
1	datasetName	managerName	num	title	
2	dataset1	Sam	9	Dev	
3					
4					
5					
6					

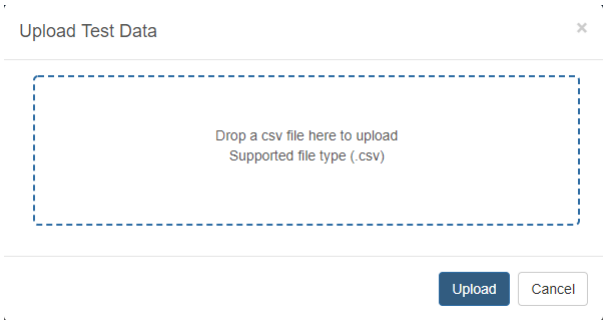
2. Save the excel file to the csv format on your local disk.

Notes:

You must use **datasetName** and **dataset1** for the first column name and value for internal usage.

Upload test case data

1. Click + button from **Test Data files** folder in the left menu sidebar.
2. Choose the csv file you want to upload.



3. After upload,new test case data files will appear on the left menu side bar.
4. You can also click download icon to download it and check the content.

Delete test case data

1. Click delete button to delete this test data.

Notes:

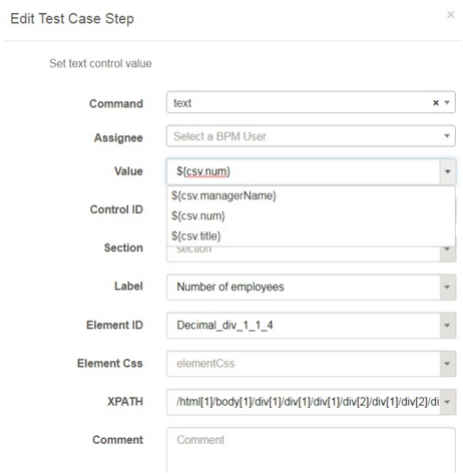
If the test case data has already bound some test cases, you need to remove the data reference of these test cases, then remove this test data.

Bind data to the test case

1. Click the test case name to see the detailed case step page.
2. Click edit button to edit this test case.
3. Bind the csv data file name from “Bind External File” selectbox.

Bind data item to the case step parameter

1. Edit the test case step which you want to use csv data.
2. Choose the csv binding variable from the selectbox value.



3. Click save button.

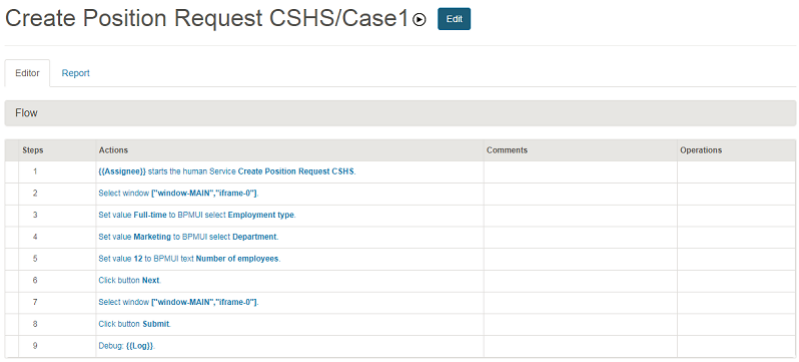
Group command management

Create a group command

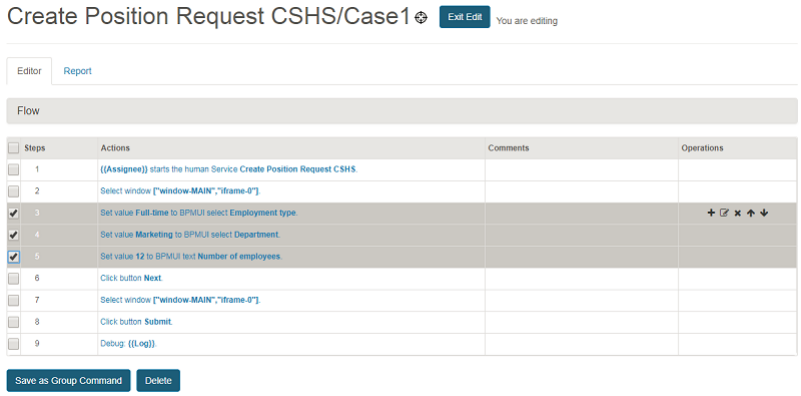
Navigate to **Hiring sample** -> **Create Position Request CSHS** -> **Case1** to generate a group command.

1. Import hiring sample project.
2. Enter project dashboard.
3. Click **Create Position Request CSHS** in left-side menu bar.

4. Click **Case1**
5. Click **Edit** button on the right of suite name.

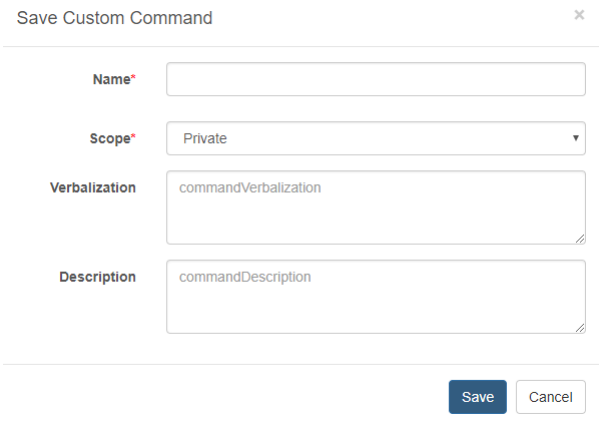


6. Choose at least two continuous case steps to save as group command.



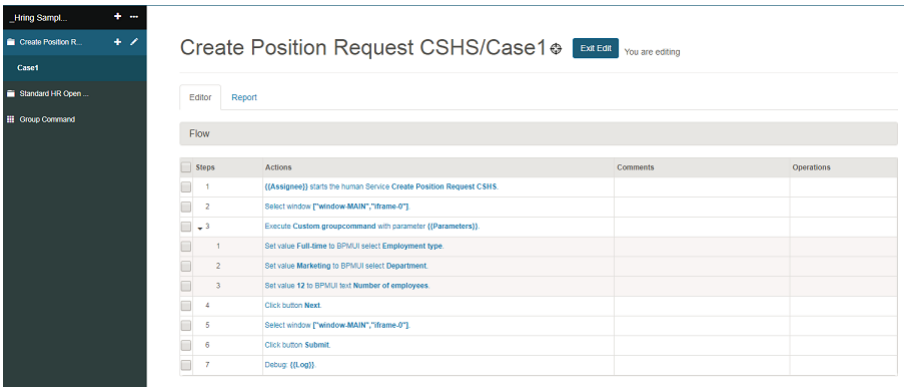
7. Click **Save as Group Command** button.
8. Fill group command name and description in the form.

   Notes The scope field has two options, the private means the command can be used only in current project, the public means the command can be used in other projects. By default the value is private.



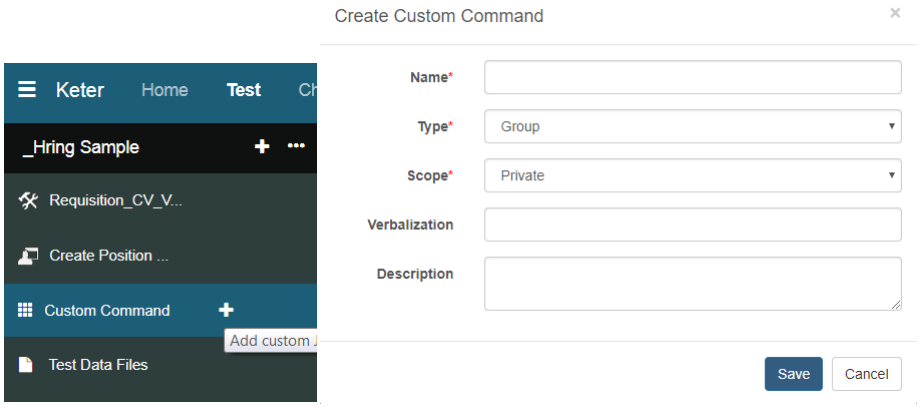
9. Click **Save** button.

   Then **Group Command** and new group command you have created appear in left-side menu bar, and the case steps you have chosen follow new group command.



10. Click **Exit Edit** button.

Notes
You can also add group command through the custom command menu.

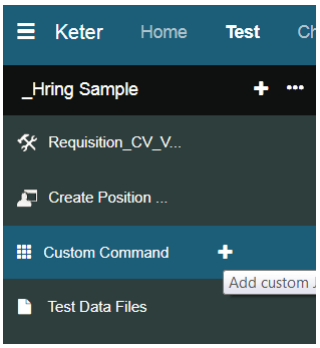


Add group command

You can add a group command when you want to add some steps which belongs to this group command.

1. Click **Edit** button on the right of suite name.
2. Click ⊞ button in the operation column of case step list table where you want to reuse a group command.
3. Fill add case step form according to below table.

   Parameter | Description
   ——————-|——————— Command | Reused group command name, in custom category Assignee | BPM user who help you finish process comment | Comment of command

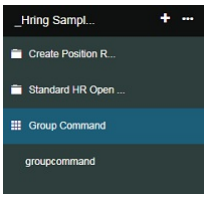


4. Click **Add** button.
5. Click **Exit Edit** button.

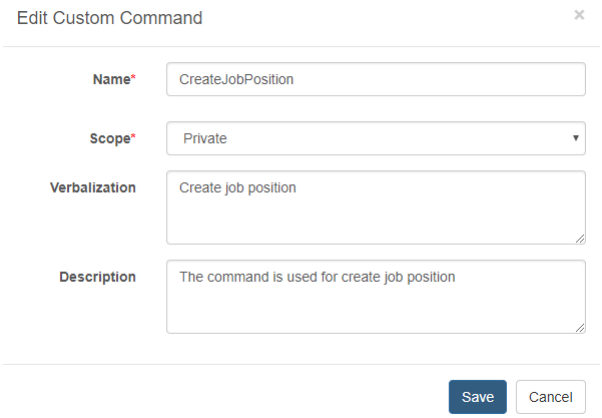
Edit group command

Edit group command name and description

1. Click **Group Command** in left-side menu bar.
2. Click ✎ button on the right of group command name.



3. Edit group command name and group command description.



4. Click **Update** button.

Edit case steps of group command

1. Click **Execute Custom._command_name_ with parameters {{Parameters}}** in case step list or click command name in left-side menu bar.
2. Edit case steps.

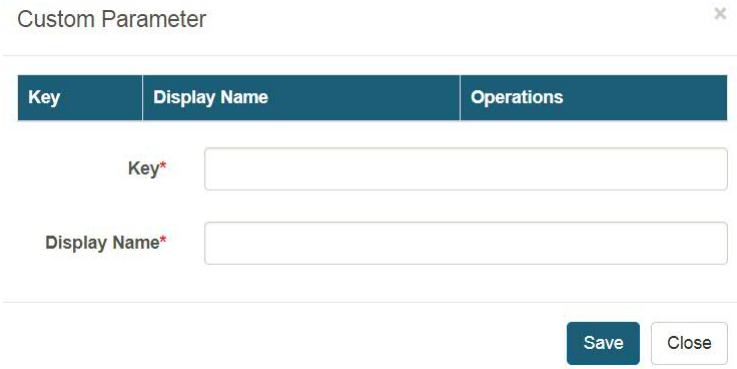
Delete Group command

1. Remove the command from test cases first.
   - Enter into case step list of every test case which includes the group command.
   - Click **Edit** button on the right of suite name.
   - Click ✖ button in group command row.
   - Click **Delete** in delete form.
   - Click **Exit Edit** button.

2. Click ✖ button on the right of group command name.

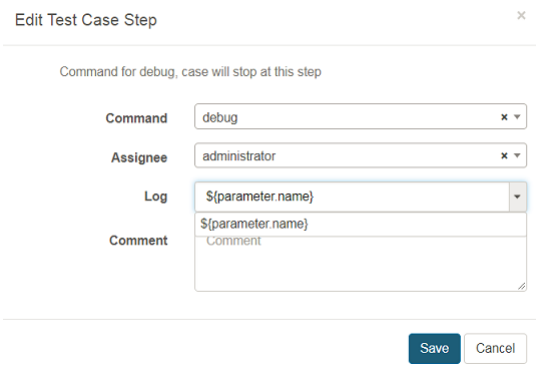
Define Group parameter

1. Define the group parameter for the group command.
   - Click **Group Command** in left-side menu bar.
   - Click **Parameter** button on the top right of group command page.
   - Enter group parameter key,display name in the group parameter page.
   - Click **Add** button to add group parameter.
   - Click **Cancel** button to exit.

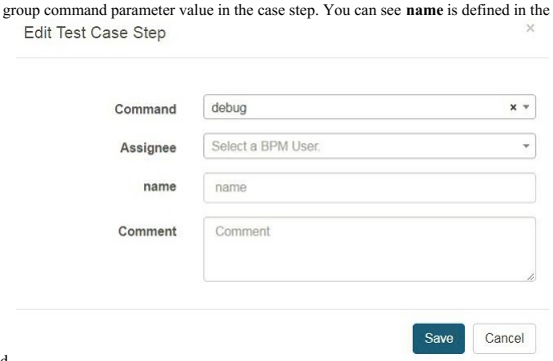


2. Bind group parameter for case step parameter.
   - Click

   button to edit case step.
   - Choose the parameter value from the select box.
   - Click **Save** button to save the parameter.



3. Edit this group command parameter value in the case step. You can see **name** is defined in the group parameter will be acted as the parameter input for group



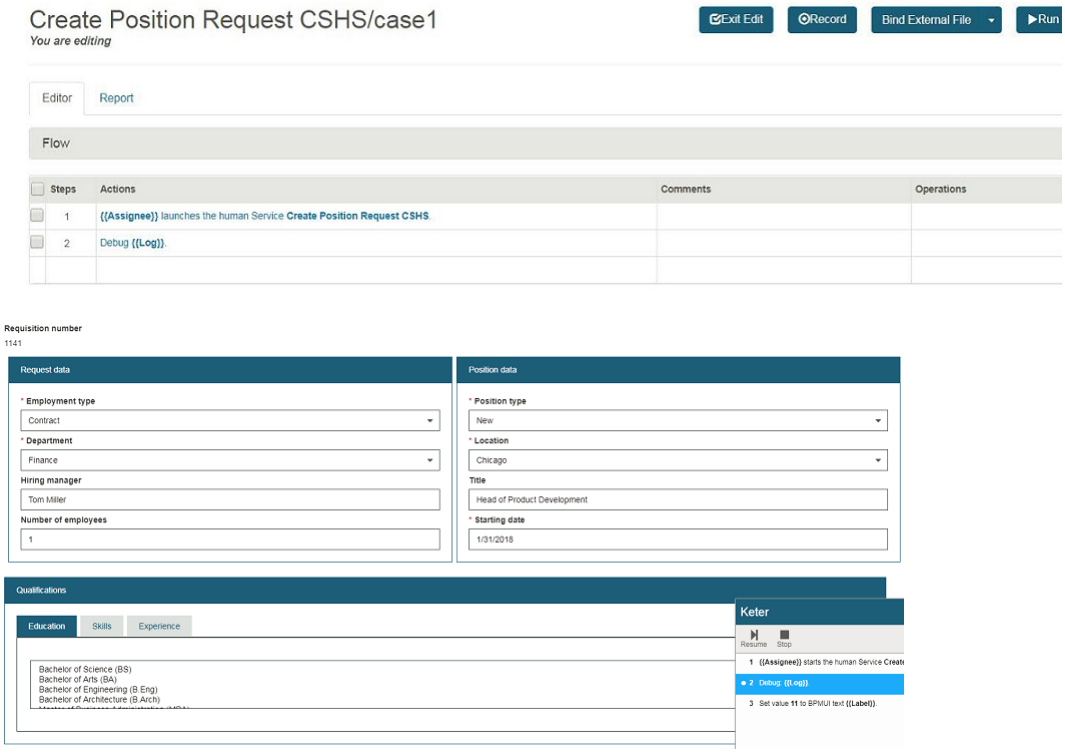
   command.

Record and replay a test case

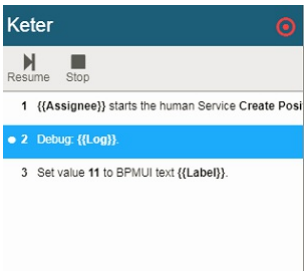
Record a test case

1. Click **Edit** button on the right of suite name.

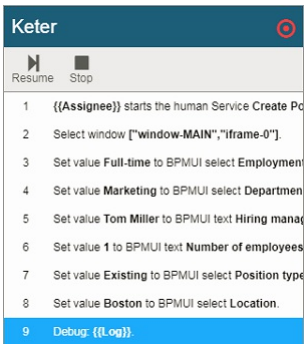
2. Click **Record** button on the right of suite name. You will enter record mode.



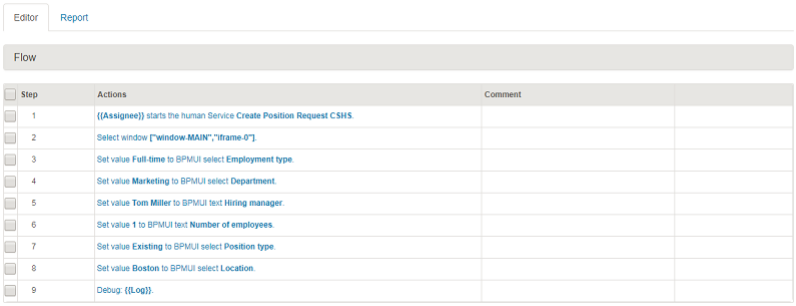
3. In the **Debug {{log}}** step, the process will be stopped. You need to click  button to resume.



4. Record according to process diagram execution flow, you can see each step you have done in tool bar.



5. Click ■ button to finish the record.The operations you done are showed in case step list table.



Replay a test case

1. Click ⊕ button on the right of case name to replay it.
2. Click ■ button to finish the replay.
3. In the **Debug {{log}}** step, the record will be stopped. You need to click ▶| button to resume.
4. Click **Exit Edit** button on the right of suite name.

Notes:

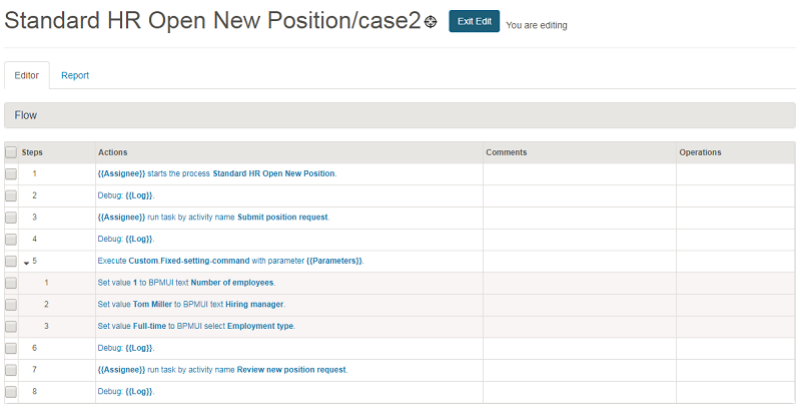
If you need to record to a test case, you can add **debug** command. When replay to the **debug** command, you can prepare to record your operation.

Record a test case with group command

1. Preparation
   - Prepare group commands that you want to add into test case. Refer to **Create a group command** in [Reusable group command management]. Here is the group command that is ready to be used as example.

   - Add group command to test case. Refer to **Reuse a group command** in [Reusable group command management].
   - Add **debug** command following the group command. In case you would like to record after the group command. Take case2 of **Standard HR Open New Position** as an example. After finishing preparation, the case step list will be like that shown in the following screenshot.

2. The recording operation refers to **Record a test case**.

Replay a test case with group command

Refer to **Replay a test case**.

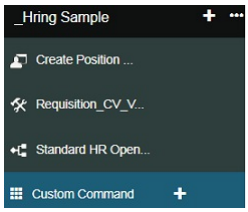
1: test-record-and-replay-a-test-case.html

title: “Write a Javascript command” category: test date: 2018-10-02 15:17:55 last_modified_at: 2019-07-29 15:17:00 order: 55 —

Write a Javascript command

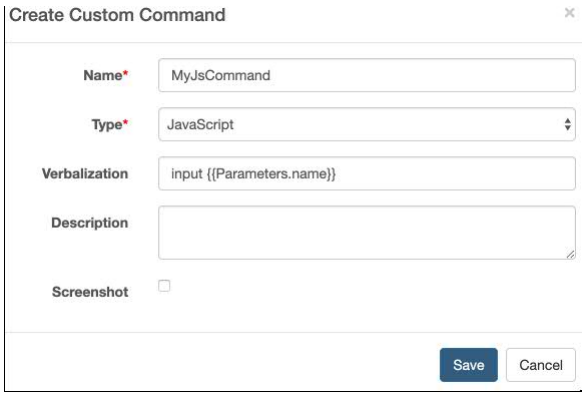
Create a javascript command

1. Navigate to **Custom Command** in the left-side menu bar, click + button to generate a group command.

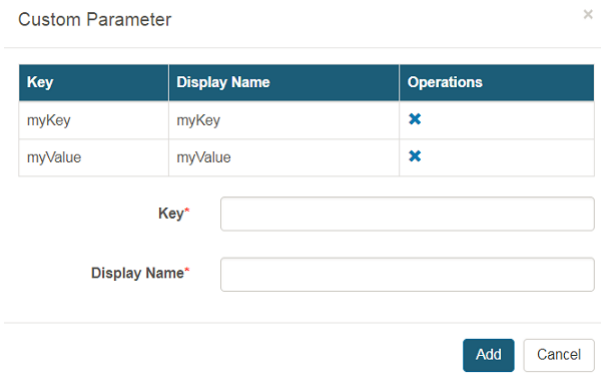


2. Enter the javascript command info.

   Notes The scope field has two options, the private means the command can be used only in current project, the public means the command can be used in other projects. By default the value is private.You can pass the {Parameters.name}} as part of verblization which you define the parameter in this js.



3. Define the javascript command parameter.

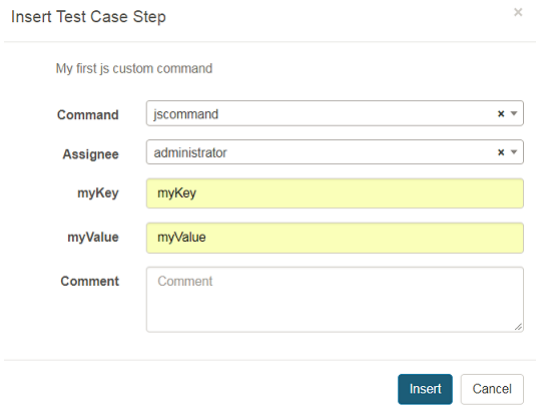


4. Define your javascript command logic.

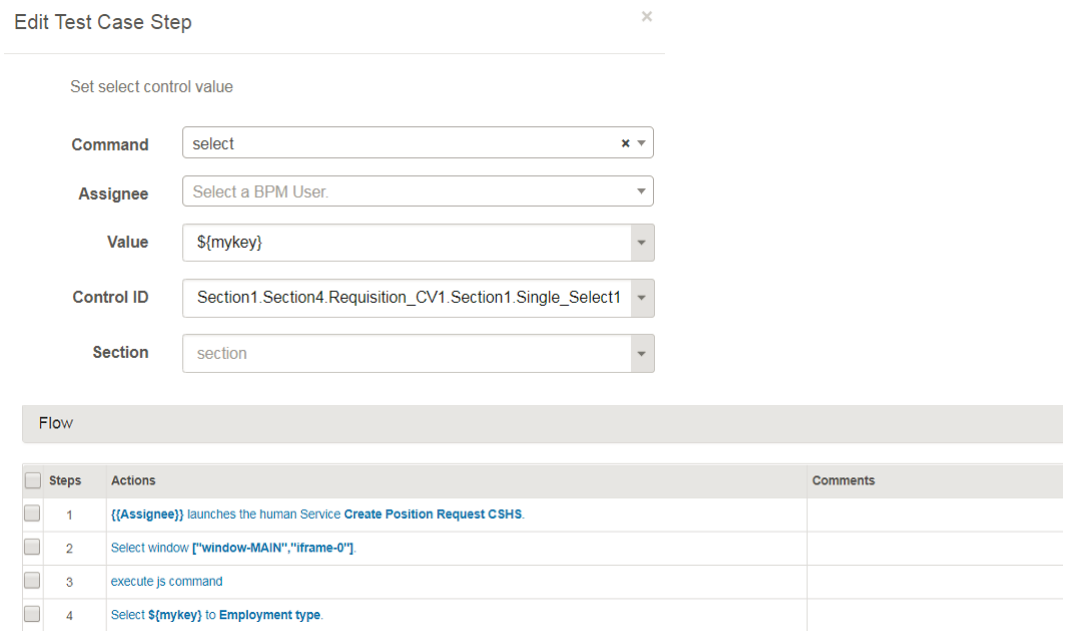
Script:

```
var myKey = this.jsonObject.myKey;
var myValue = this.jsonObject.myValue;

var parameter = {
  key : myKey,
  value: myValue
};

return bta.util.callback("putContext", parameter);
```

5. Add your javascript command in your test case.



6. You can reference js variable in the case step



7. Refer to Javascript API Reference for OOTB Javascript functions

Javascript Command sample

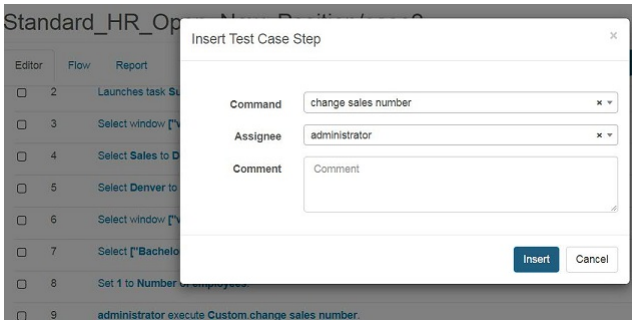
In Hiring sample Standard_HR_Open_New_Position/case2,we will create a addition logic if choose position Request department is **Sales**, we will change the number of employees to value 3 no matter the value we choose before.

1. We create a js testCommand name **change sales number**
2. Copy the below js command content into this js command.

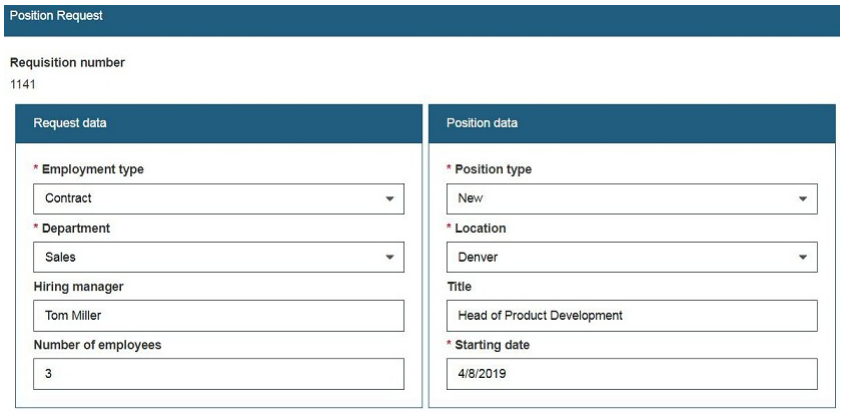
```
var controlId='Section1.Section4.Requisition_CV1.Section1.Single_Select2';
var textId='Section1.Section4.Requisition_CV1.Section1.Text3';
```

```
var coachView;
var inputElement;
var selectedValue = [];
if (controlId) {
    coachView = bta.util.getCoachViewByControlId(controlId);
}
var selectElement;
if (coachView) {
    selectElement = coachView.context.element.querySelector("select.form-control");
    if (bta.util.isElementTag(selectElement, "select")) {
        var selectOptions = selectElement.options;
        for (var i=0; i<selectOptions.length; i++) {
              if (selectOptions[i].selected) {
                 if (selectOptions[i].text=='Sales'){
                    var textView = bta.util.getCoachViewByControlId(textId);
               inputElement = textView.context.element.querySelector("input.form-control");
                    if (inputElement) {
                      inputElement.value=3 ;
                      }

              }
                 }
            }
        }
}
return bta.util.callback();
```

3. Insert this js command in the **Submit position request** task UI.



4. When you record/replay this case again,you will find when you recorded department value is **sales** ,the number of employee value is **3**.



Write a Java command

Software Prerequisites

- Eclipse IDE
- Maven

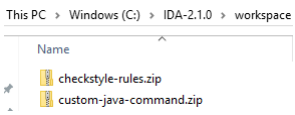
Prerequisite

You have installed **Eclipse IDE** and **Maven** in your local environment.

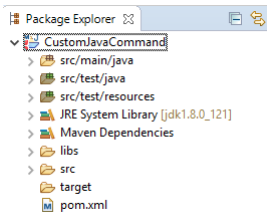
Java command customization

Import CustomJavaCommand Java project

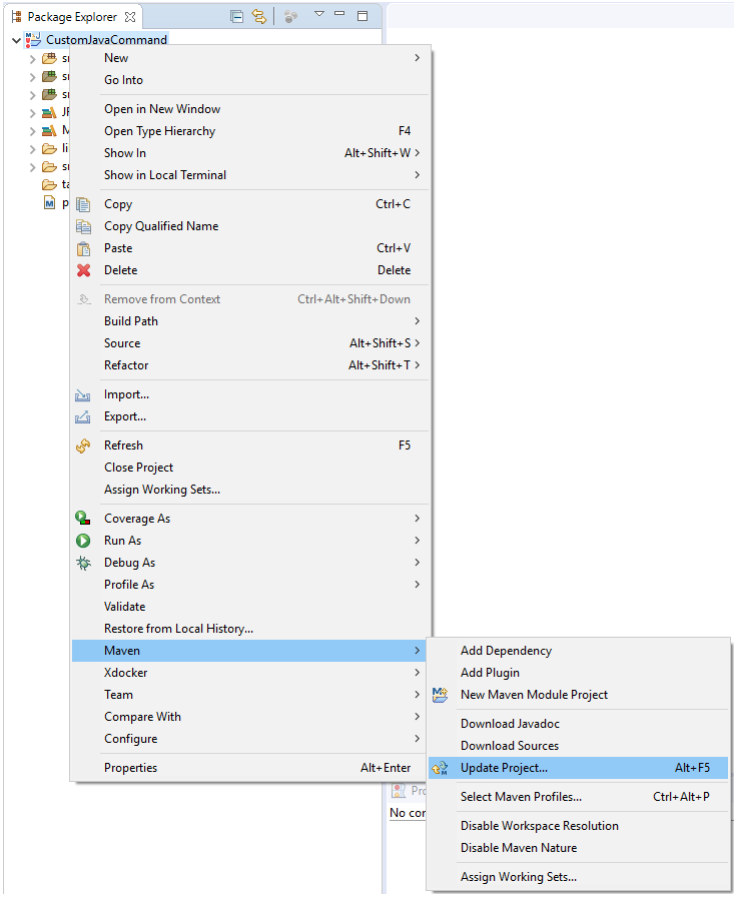
1. Unzip the file **workspace/custom-java-command.zip** in release package.



2. Import the project **CustomJavaCommand** into **Eclipse**.



3. Perform **Maven Update** to load all dependency libraries.



Write custom Java command

1. Create a Java class **MyJavaCommand** under the source folder **src/main/java**.

```
public class MyJavaCommand {

}
```

2. Add annotations **@CustomCommand** and **@Parameter** on the class, so that the IDA can recognize the Java command. And add private variables based on the parameter definition.

   @CustomCommand

   - **name** : The name shows on the test command selection box.
   - **verbalization** : The text shows on the test case detail table.
   - **description** : The text shows on the add/update command dialog.
   - **takeScreenshot** : Indicate whether it will take screenshot after command execution.

   @Parameter

   - **name** : The name to get the parameter from Java code.
   - **displayName** : The name shows on the add/update command dialog.
   - **type** : The control type shows on the add/update command dialog. It supports four kinds of type: text, textarea, combo and checkbox.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import com.ibm.cte.sdc.bpm.testing.command.annotation.CustomCommand;
import com.ibm.cte.sdc.bpm.testing.command.annotation.Parameter;

@CustomCommand(name = "myJavaCommand", verbalization = "Hello {{Parameters.message}}", description = "execute my java command",
takeScreenshot = true)
@Parameter(name = "message", displayName = "Message", type = "text")
@Parameter(name = "value", displayName = "Value", type = "text")
public class MyJavaCommand {

  private static final Logger logger = LoggerFactory.getLogger(MyJavaCommand.class);

  private String message;
```

```
  private String value;

}
```

3. Extend the Java command from class **com.ibm.cte.sdc.bpm.testing.command.BaseCommand**, and override the constructor method and the method **execute** and **parseArgs** from parent class as below:

```
  public MyJavaCommand(TestDriver driver) {
    super(driver);
  }

  @Override
  public Map<String, Object> execute(Map<String, Object> args) throws Exception {
    if (!parseArgs(args)) {
      logger.error("Can not execute MyJavaCommand due to invalid args {}", args);
      this.assertFail("Args invalid", false);
    }

    //TODO add your execution logic

    return this.result;
  }

  @Override
  protected boolean parseArgs(Map<String, Object> args) {
    super.parseArgs(args);
    this.message = (String) args.get("message");
    this.value = (String) args.get("value");
    boolean valid = true;
    if (StringUtils.isBlank(message)) {
      logger.error("Wrong args with message {}", message);
      valid = false;
    }
    return valid;
  }
```

4. By overriding the **parseArgs** method, you could read the inputs from a **Map** object and assign the value to the pre-defined private variables. You could also add your own validation logic in the method, and return **false** if the arguments is invalid.

5. By overriding the **execute** method, you could add your own execution logic. An example of **execute** method as below:

```
  @Override
  public Map<String, Object> execute(Map<String, Object> args) throws Exception {
    if (!parseArgs(args)) {
      logger.error("Can not execute MyJavaCommand due to invalid args {}", args);
      this.assertFail("Args invalid", false);
    }

    // The driver object can handle the Browser UI, eg: open a url
    this.driver.open("about:blank");

    // Invoke BPM Server REST API.
    TeamworksAPI client = getTeamworksAPI();
    client.getSystemDetails();

    // Put variable into context, so that the variable can be used in other commands.
    // We need to use the variable by the format ${context.JAVA_COMMAND_VARIABLE}
    this.result.put("JAVA_COMMAND_VARIABLE", this.value);
    // Log
    logger.info("Hello {}", message);

    return this.result;
  }
```

   - **this.driver**: It is the object can handler browser UI.
   - **this.getTeamworksAPI()**: This method return an object which can invoke the BPM server REST API.
   - **this.result**: The result of the Java command, and the values in the result will pass to the following test commands in a test case.

6. Below is a completed example of Java command, you could also found it in the package **com.ibm.cte.sdc.bpm.testing.command.customized.MyJavaCommand**

```
package com.ibm.cte.sdc.bpm.testing.command.customized;


import java.util.Map;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.ibm.cte.sdc.bpm.testing.command.BaseCommand;
import com.ibm.cte.sdc.bpm.testing.command.annotation.CustomCommand;
import com.ibm.cte.sdc.bpm.testing.command.annotation.Parameter;
import com.ibm.cte.sdc.bpm.testing.driver.TestDriver;
import com.ibm.websphere.bpm.api.TeamworksAPI;
import com.ibm.websphere.bpm.util.StringUtils;

@CustomCommand(name = "myJavaCommand", verbalization = "Hello {{Parameters.message}}", description = "execute my java command",
takeScreenshot = true)
@Parameter(name = "message", displayName = "Message", type = "text")
@Parameter(name = "value", displayName = "Value", type = "text")
public class MyJavaCommand extends BaseCommand {

  private static final Logger logger = LoggerFactory.getLogger(MyJavaCommand.class);

  private String message;
  private String value;

  public MyJavaCommand(TestDriver driver) {
    super(driver);
  }

  @Override
  public Map<String, Object> execute(Map<String, Object> args) throws Exception {
    if (!parseArgs(args)) {
      logger.error("Can not execute MyJavaCommand due to invalid args {}", args);
      this.assertFail("Args invalid", false);
    }

    // The driver object can handle the Browser UI, eg: open a url
    this.driver.open("about:blank");

    // Invoke BPM Server REST API.
    TeamworksAPI client = getTeamworksAPI();
    client.getSystemDetails();

    // Put variable into context, so that the variable can be used in other commands.
    // We need to use the variable by the format ${context.JAVA_COMMAND_VARIABLE}
    this.result.put("JAVA_COMMAND_VARIABLE", this.value);
    // Log
    logger.info("Hello {}", message);

    return this.result;
  }

  @Override
  protected boolean parseArgs(Map<String, Object> args) {
    super.parseArgs(args);
    this.message = (String) args.get("message");
    this.value = (String) args.get("value");
    boolean valid = true;
    if (StringUtils.isBlank(message)) {
      logger.error("Wrong args with message {}", message);
```

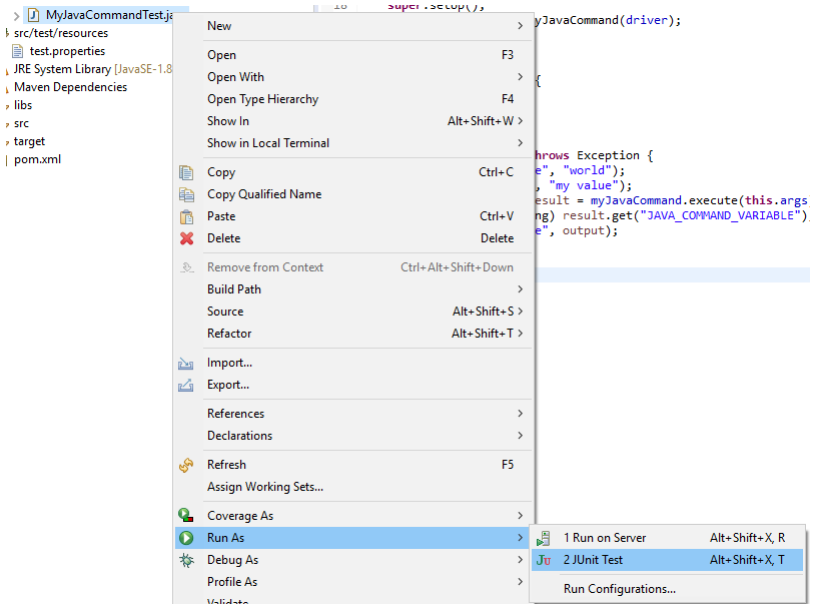
```
      valid = false;
    }
    return valid;
  }
}
```

Unit test custom Java command

1. Configure **test.properties** under **src/test/resources**.
   - **selenium_hub_url**: It's the selenium grid hub url. If you don't have selenium grid, then you can setup one by the link.
   - **selenium_hub_browser**: The browser type, the value could be **iexplore**, **firefox** or **chrome**, please make sure the selenium hub can support the configured browser type.
   - **bpm_server_url**: The BPM server URL
   - **bpm_server_user_name**: The BPM REST call user name
   - **bpm_server_password**: The BPM REST call password
2. Write unit test class under the source folder **src/test/java**, an example as below:

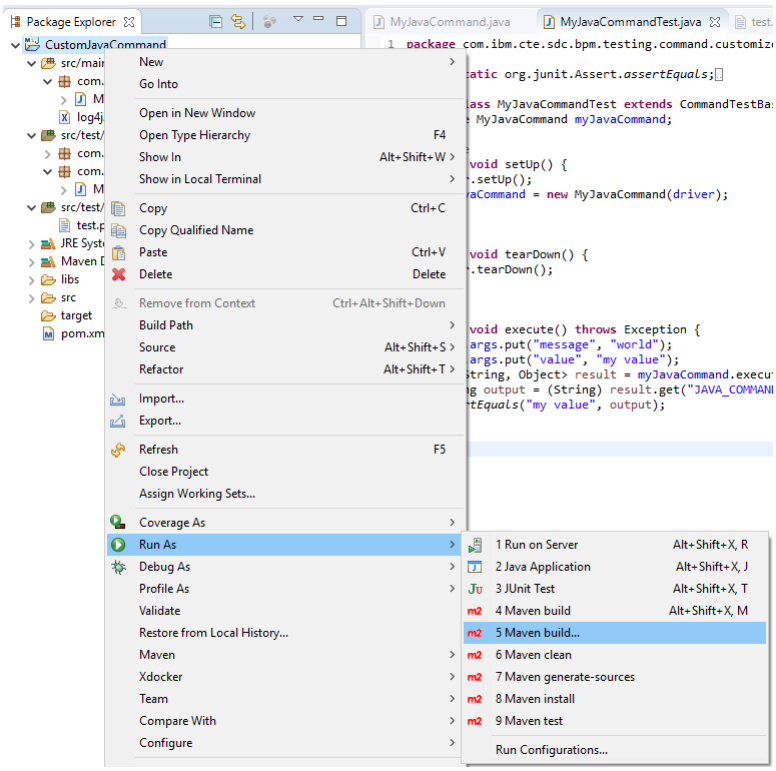
```
package com.ibm.cte.sdc.bpm.testing.command.customized;

import static org.junit.Assert.assertEquals;

import java.util.Map;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import com.ibm.cte.sdc.bpm.testing.command.CommandTestBase;

public class MyJavaCommandTest extends CommandTestBase {
  private MyJavaCommand myJavaCommand;

  @Before
  public void setUp() {
    super.setUp();
    myJavaCommand = new MyJavaCommand(driver);
  }

  @After
  public void tearDown() {
    super.tearDown();
  }

  @Test
  public void execute() throws Exception {
    this.args.put("message", "world");
    this.args.put("value", "my value");
    Map<String, Object> result = myJavaCommand.execute(this.args);
    String output = (String) result.get("JAVA_COMMAND_VARIABLE");
    assertEquals("my value", output);
  }
}
```

3. Run the class as **JUnit Test**

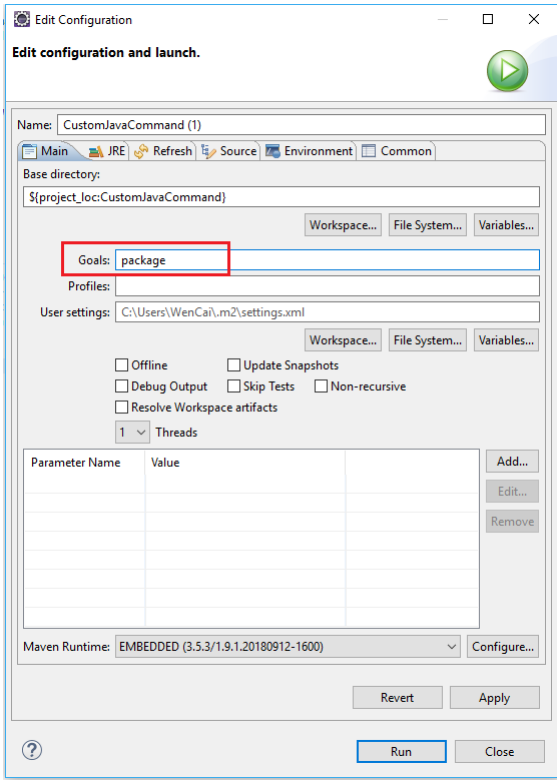


Package custom Java command

1. Once all unit test cases are passed, then you could package the Java command by right-click the project and choose **Run As** -> **Maven build…**.



2. In the Maven build configuration page, enter **package** in the textbox **Goals**, then click **Run** button.

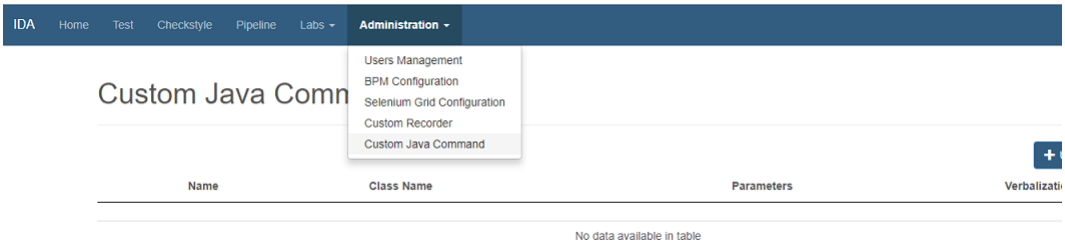


3. You could find the Jar file at the path **target/custom-java-command-1.0.jar**, this is the jar we will upload to IDA application.

- Notes: If you use any java lib which are not existed in as-is IDA.war/lib folder, you need copy it to lib folder and repack the IDA web so that the customer java command jar could find reference in class loader.

Upload custom Java command in IDA

1. Open the page in **Administration** -> **Custom Java Command**.

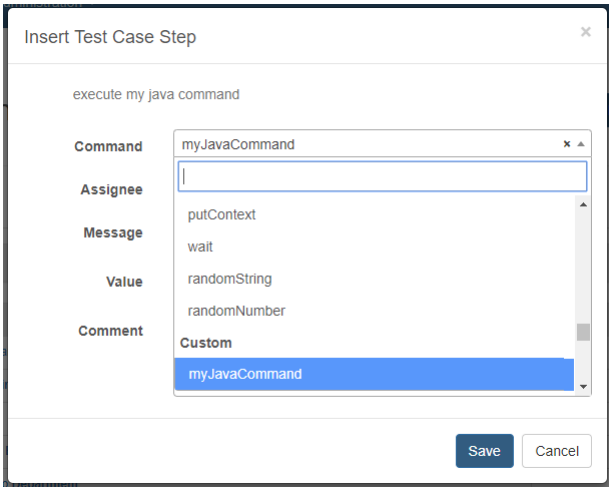


2. Click **Upload** button, choose the file **custom-java-command-1.0.jar** and upload it.

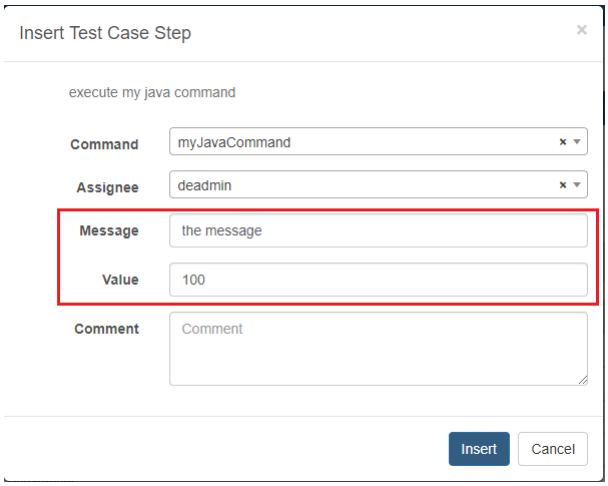
3. You could see the Java command shows in the table.

Use custom Java command in IDA test case

1. Edit your test case, and open the add/update command dialog. You will see the Java command shows under the **Custom** category.



2. The parameters defined by Java annotation are also shows on the command dialog.



Test Smart Comparison

How does smart comparison work

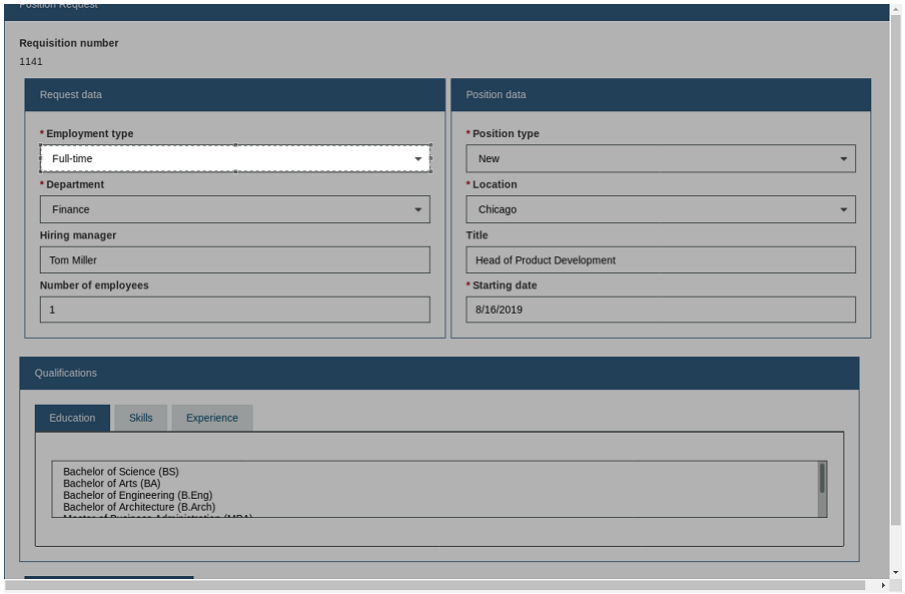
Smart comparison is an image assertion. Smart comparison works on case steps which have screenshot in test report. It compares current screenshot with the saved screenshot in smart comparison. If result is **MISMATCH**, testing result of this case step is **Failed**, reports *Screenshot does not match the expected image in smart comparison* error.

Start Smart Comparison

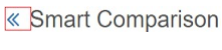
Notes: Before starting smart comparison, please make sure that the test case already finish running and has a report. The status of the report is **Passed** and there is at least one case step with screenshot.

1. Go to test case edit page.
2. Click **Edit** button.
3. Click **Smart Comparison** Button. You can see that all comparable case steps of this test case are listed on the left. And click one, its screenshot shows on the right.
4. Click

in front of the case step to enable image comparison. The whole image will be compared as default. If you don't want to compare the whole image, you can specify the area to compare by moving the mouse over the screenshot, then clicking, dragging and select the area you want to compare. Release the mouse and can see the selected part is shown:

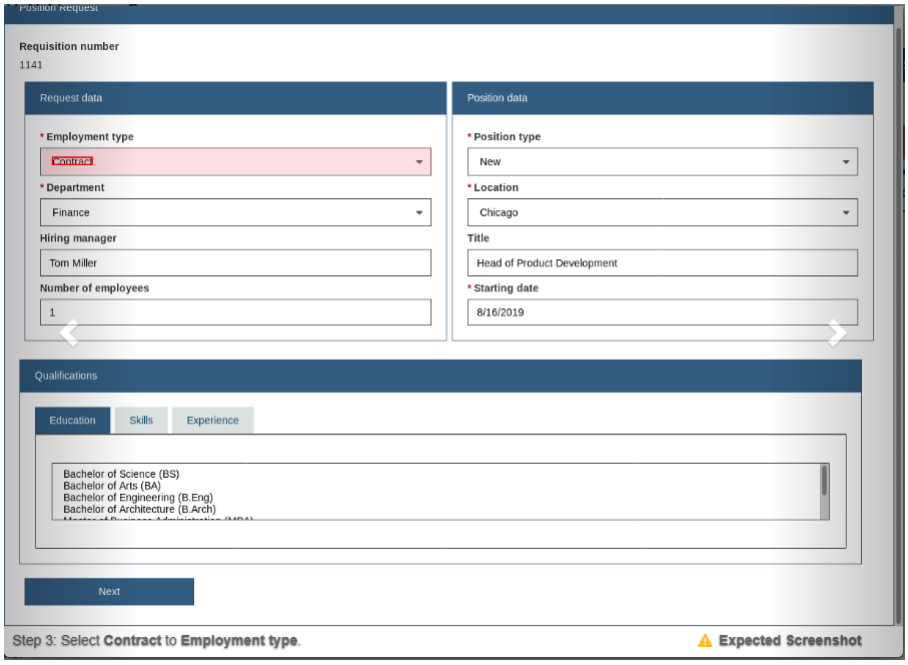


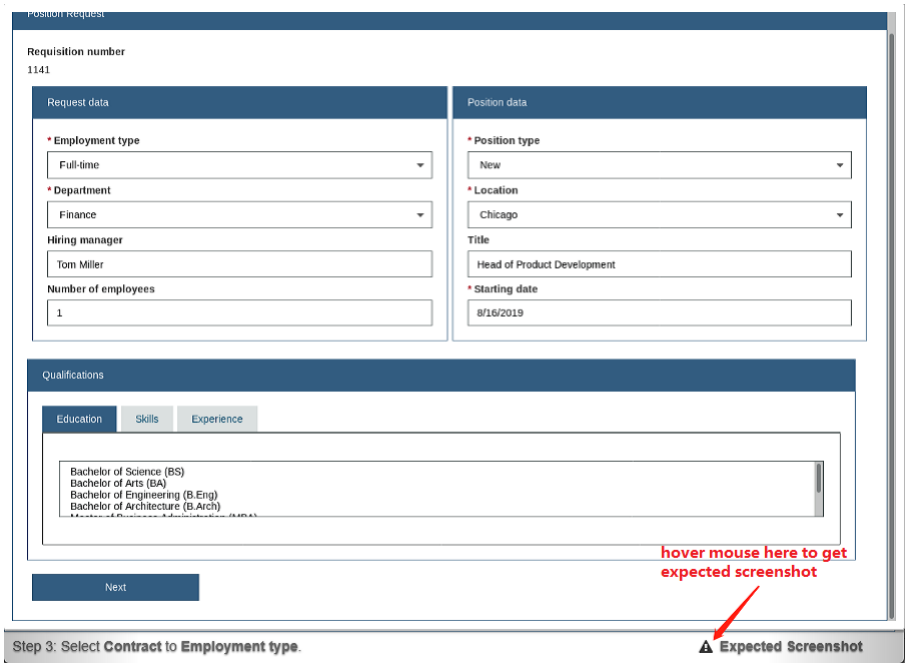
5. If you want to unselect the selected area, click the other area. The confirm form will be shown, and click **OK**. It will switch back to the whole image comparison.
6. Click << button and you can go back to test case edit page.



7. Click **Run** button and wait for report.
8. The comparison result is shown in the report. If **MISMATCH**, the trace:

9. Click the screenshot, the comparison area will be set with different background color, and the differences between the expected image has been set in smart comparison page and current screenshot will be highlighted in red line. Hover the mouse over the **Expected Screenshot** can get the expected screenshot :





If there is nothing different, current screenshot will be shown and status is **Passed**.

10. Click in front of the case step again to disable image comparison.

Fetch Screenshots from Latest Report

Click **Refresh** button in Smart Comparison page can get screenshots from latest report, and will reset all settings.

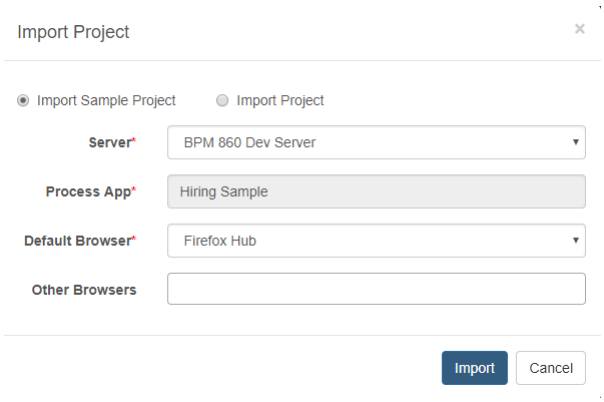
Notes:

The **Refresh** button is only displayed when the status of latest report is **Passed** and there is at least one case step with screenshot in this test case.

Import and execute sample test project

Import a sample test project

1. Click the **Sample Project** button in the project tab.

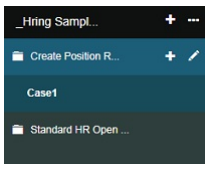


2. Fill import form according to following table.

Parameter	Description
Server	Server environment
Process App	Process app list of the selected server, default value is “Hiring Sample”
Selenium Hub	Support chrome, firefox selenium node
BPMoc	Using BPMoc check

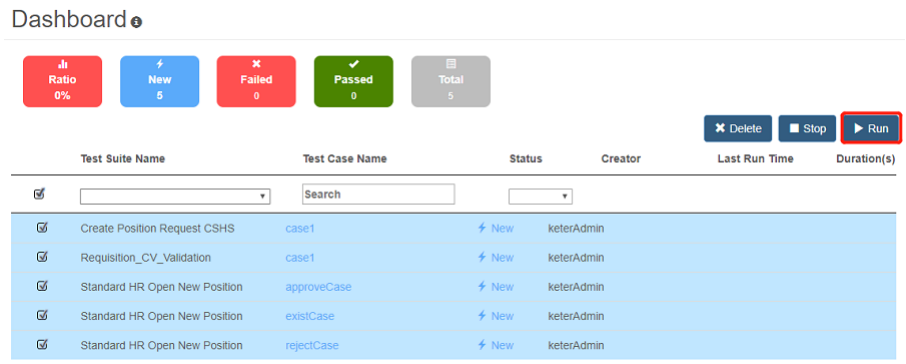
3. Click **Import** button.
4. Click **_Hring Sample** from project list and enter into project dashboard.

The default sample project contains two suites: **Create Position Request CSHS** and **Standard HR Open New Position**.

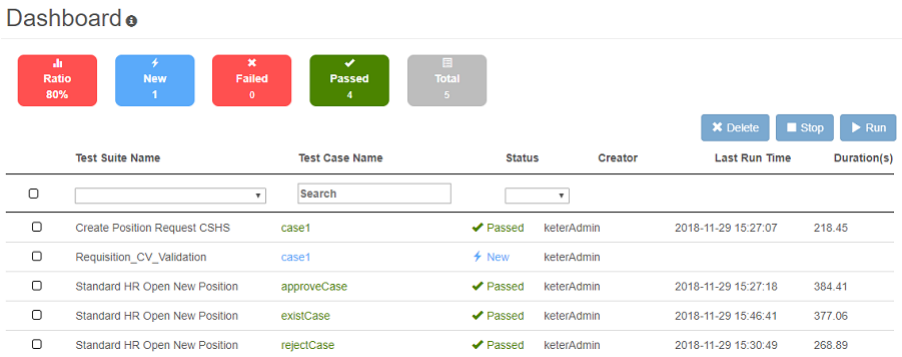


Execute a sample test project

1. Select all the test cases in the dashboard and click **Run** button.



The test result and basic information of hiring sample project can be viewed in project dashboard.



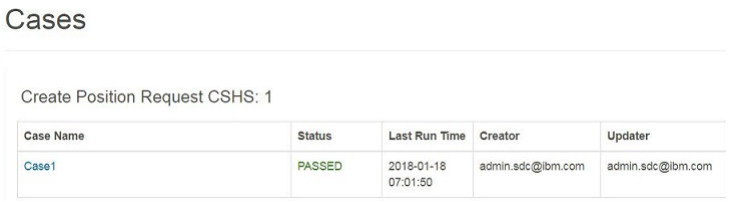
2. You can see report of a test case by clicking case name ,then clicking **report**.More details.

Test Report

IDA offers detailed reports including screenshots, execution process flow and coverage to improve communication between testers and developers on every test case.

Test Case Report

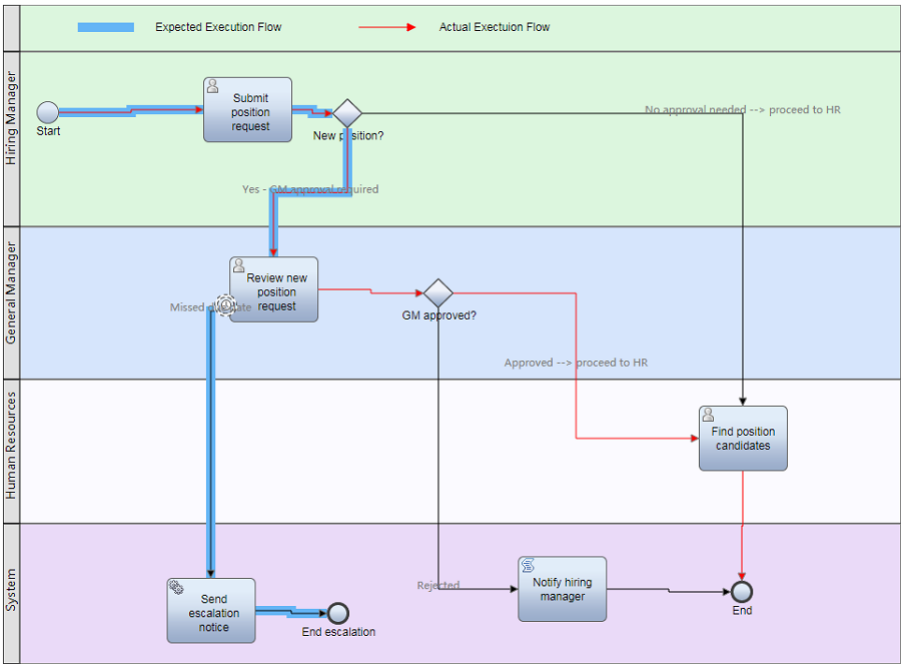
In the Test project dashboard page,you can click any test case link to see the detail case report.



Switch the **Report** tab ,check the every step of execution status.
Click the **screenshot** image link to see the screenshot if this case step includes UI.

In process diagram execution flow, You can see two lines.

The **blue** line stands for the expected process execution flow.
The **red** lines stands for the acutal execution flow.
You can adjust your test case based on the difference between “acutal flow” and “expected flow” process diagram.

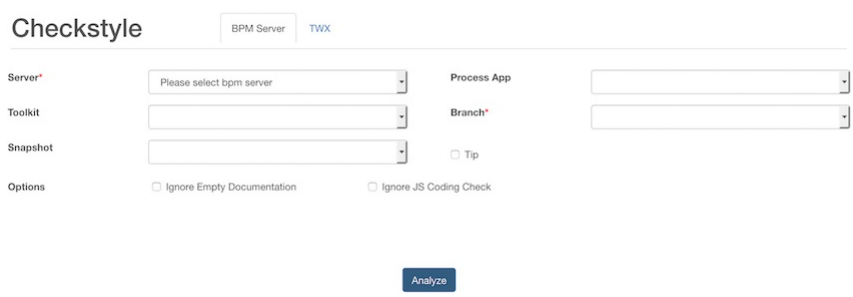


Analyze BPM projects with checkstyle

Setting up the BPM project and perform it.

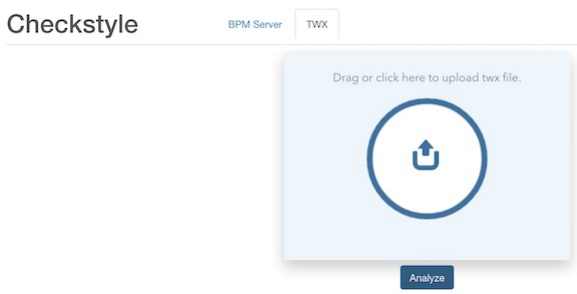
1. Use BPM server or upload a TWX file to analyze.

   Use BPM server, fill the below field values for the checkstyle form.

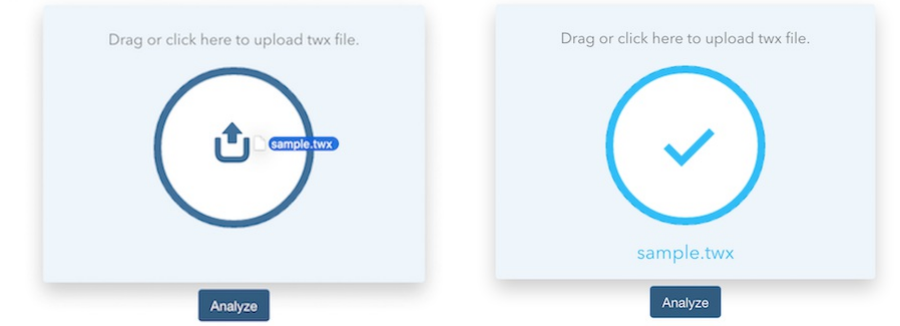


Parameter	Description
Server	Server environment
Process App	Process app list on the selected server
Toolkit	Toolkits of selected server
Branch	The branch of selected process app or selected toolkit
Snapshot	The snapshot of selected branch
Tip	The tip snapshot of the process app
Ignore Empty Documentation	Ignoring empty documentation check
Ignore JS Coding Style Check	Ignore JS coding style check check

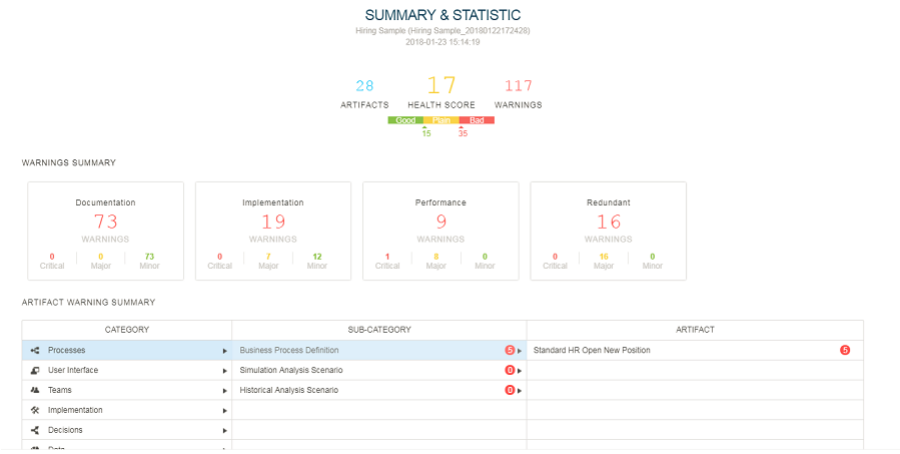
   Upload a TWX file, choose the TWX tab.



You can choose or drop a TWX file.



2. Click **Analyze** Button to see the CheckStyle report.



Customize Checkstyle rules

Software Prerequisites

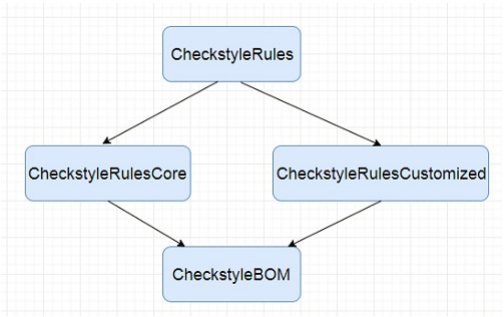
- IBM® Operational Decision Manager

Prerequisite

You have installed **Rule Designer** in your local environment and **Checkstyle Rule Projects** have been in your local workspace.

BPM checkstyle rule projects overview

The BPM checkstyle rule projects include **CheckstyleBOM**, **CheckstyleRules**, **CheckstyleRulesCore** and **CheckstyleRulesCustomized**. Their dependency relationships are as follows:

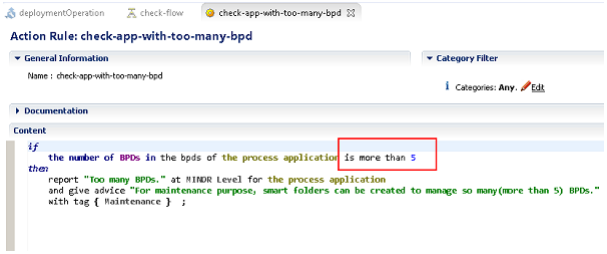


BPM checkstyle rules customization

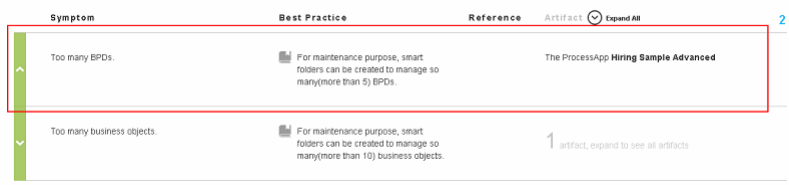
Override core checkstyle rules

To rewrite any core checkstyle rule, please do not update the core rule itself. Instead you should copy the rule into the **checksOverride** rule package of project CheckstyleRulesCustomized and then update the copy. Take overriding a core checkstyle rule **check-app-with-too-many-bpd** as an example:

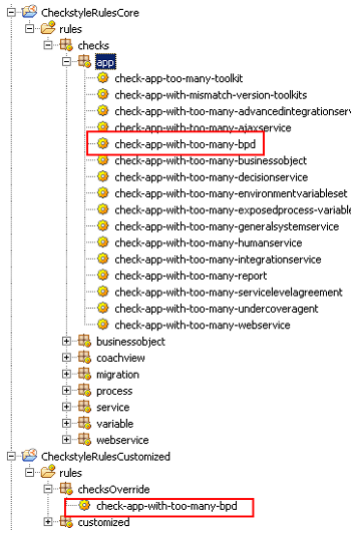
1. Review the content of the core rule **check-app-with-too-many-bpd**. if the process app has more than 5 bpds, then a checkstyle warning message will be generated.



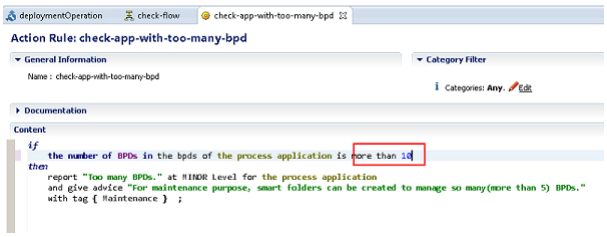
2. Run the BPM checkstyle asset against an updated sample file **sample-updated.twx** and verify in the final checkstyle report, there is warning message **Too many BPDs**.



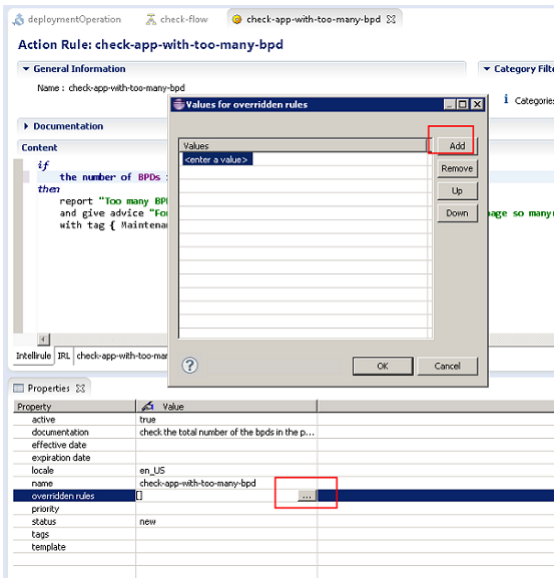
3. In rule designer, copy the rule **check-app-with-too-many-bpd** from rule project CheckstyleRulesCore to rule package checksOverride in rule project CheckstyleRulesCustomized.

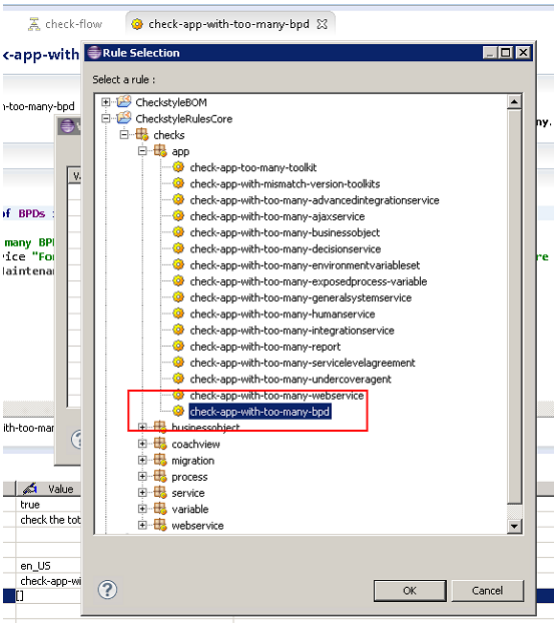


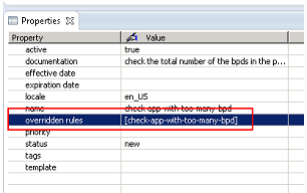
4. Update the rule **check-app-with-too-many-bpd** in CheckstyleRulesCustomized, change number 5 to 10.



5. In the properties window of the rule **check-app-with-too-many-bpd**, select the core rule (with the same name) to override.



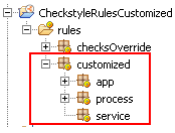




6. Redeploy the ruleset to your target server e.g. on IBM Cloud.
7. Rerun BPM checkstyle asset against an updated sample file **sample-updated.twx** and verify in the final checkstyle report, there is **no** warning message **Too many BPDs**.

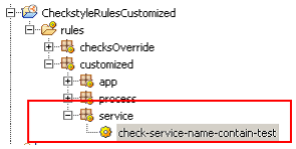
Write custom checkstyle rules

To write customized checkstyle rules, put the new rules in rule package **customized** in rule project **CheckstyleRulesCustomized**. Create artifact-based rule package if necessary.

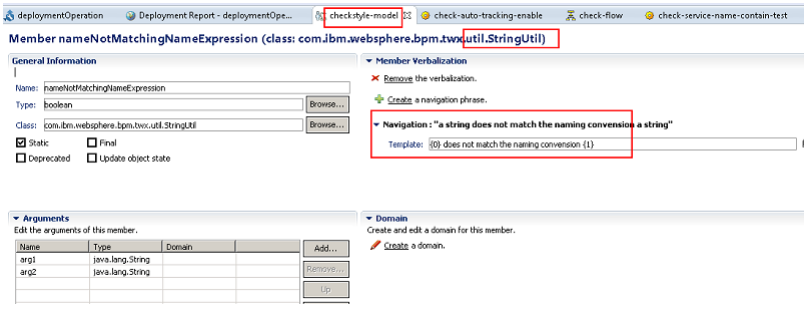


Take writing a customized checkstyle rule **check-service-name-contain-test** as an example:

1. Add a new rule **check-service-name-contain-test** that will check if a BPM service name contains test into in package **customized** in rule project **CheckstyleRulesCustomized**.

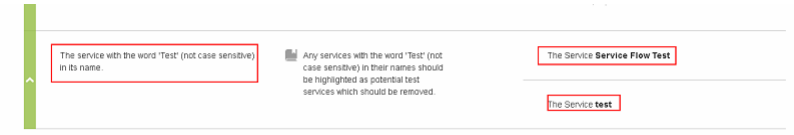


2. Check that the asset provides the utility function **does not match the naming convension** in its **CheckstyleRulesBOM** project.



3. Redeploy the ruleset to your target server e.g. on IBM Cloud.

4. Rerun BPM checkstyle asset against an updated sample file **sample-updated.twx** and verify in the final checkstyle report, there is a warning message **The service with the word 'Test' (not case sensitive) in its name**.



Checkstyle report

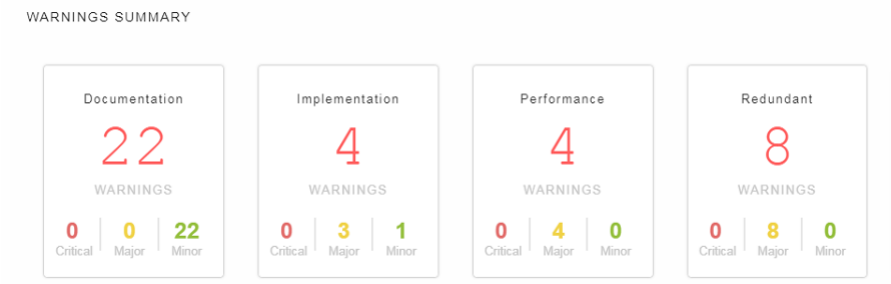
Summary & Statistic

Warnings summary

After finishing the analysis, a comprehensive mark named **HEALTH SCORE** would be given in this section to evaluate the selected process application's code is good or bad according the coding standard. More details, see the following screenshot.

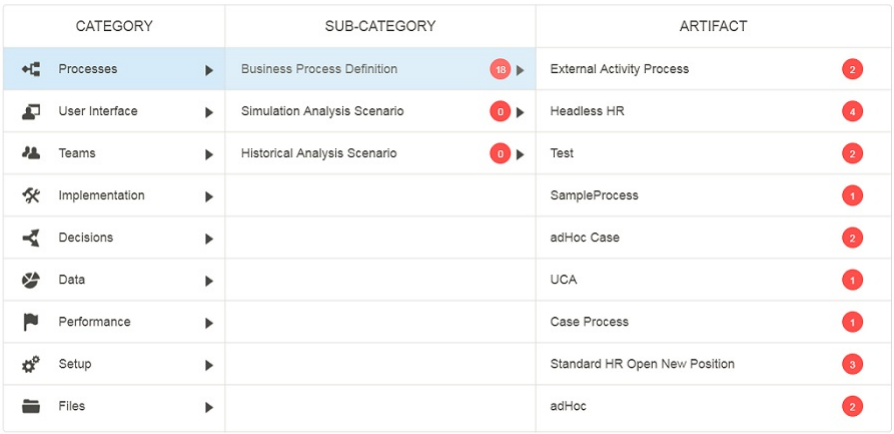
Lower score means that the code of this process application is closer to to the coding standard, and higher score means that the code needs to refactor.

The warnings are calculated from **Documentation**, **Implementation**, **Maintenance**, **Migration**, **Performance**, **Redundant** etc. dimensions and the dimension will not be displayed when there is no warning that belongs to this dimension. Warnings of each dimension will be counted in three level : **critical**, **Major** and **Minor**. Developers can find the modification advises for each BPM application artifacts, so that it can continuously improve the coding quality.



Artifact Warning

Summary In this part, it makes statistics based on dimensions of category, sub-category and artifact. You can click **CATETORY** column to see its sub-category, click **SUB-CATEGORY** to show its artifacts and the number in the red circle is the number of warnings that sub-category or artifact you select. Click **ARTIFACT** entry to get warning details.(Refer to **Check Result Details**)

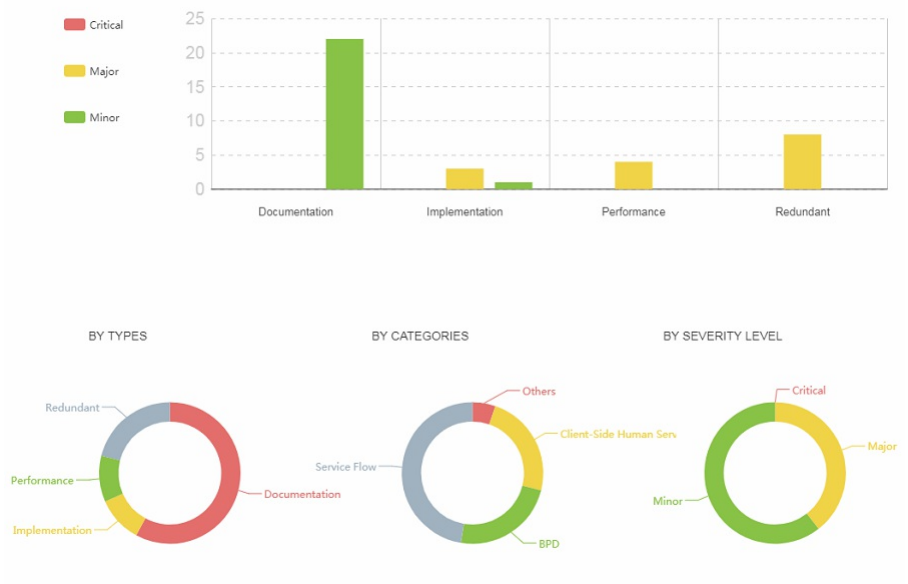


Basic Info

The basic information for this application will be displayed in this section, including dependencies and BPMN artifact summary. The number on the right of the BPMN artifact summary indicates the number of applications applied to the left artifact. More details, see the following screenshot.

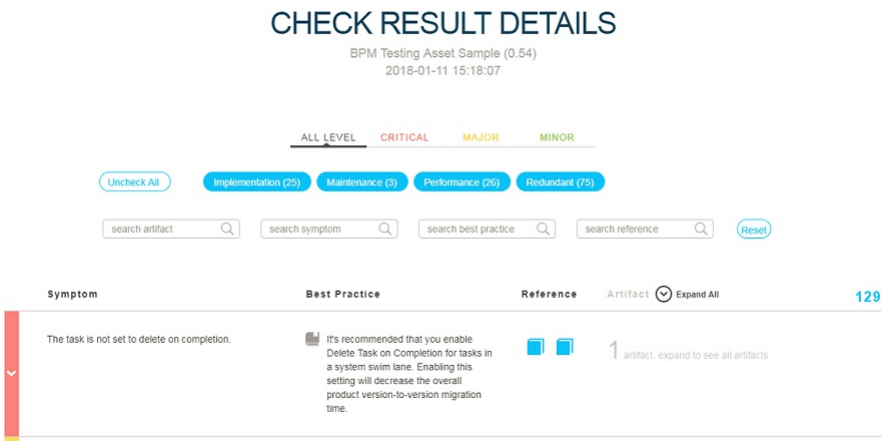
Analysis

This section is the pie chart and bar chart of the **SUMMARY & STATISTIC** part that calculated from different dimensions. More details, see the following screenshot.



Check result details

You can see more detailed information about checkstyle warnings. It supports to filter through different dimensions like level and category. You can type in **Symptom**, **Best Practice**, **Reference** and **Artifact** to make a more accurate search. **Symptom** provides the description of warnings, **Best Practice** offers suggestions for improvement, **Reference** gives a reference source for suggestions, and **Artifact** indicates the location of artifact that causes warnings so that you can fix them easily.



Deploy Checkstyle ruleset

Software Prerequisites

- IBM® Operational Decision Manager

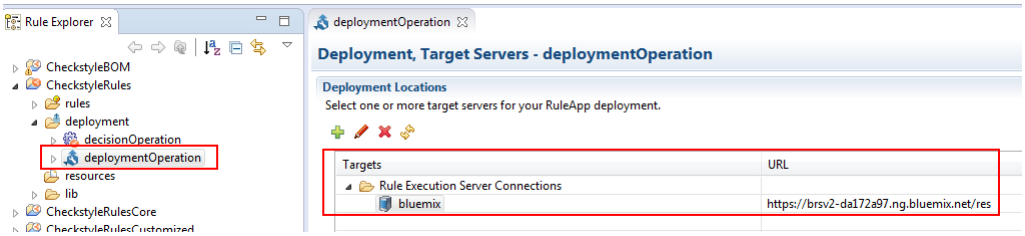
Deploy ruleset to IBM® Operational Decision Manager Decision Server

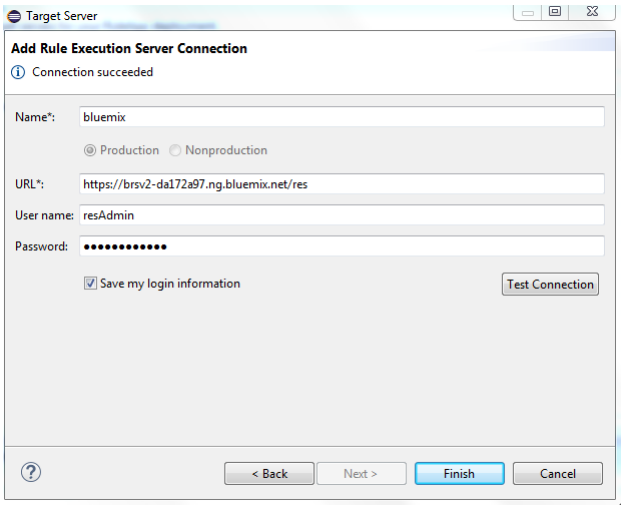
If you want to execute ruleset through Decision Server, you should deploy your ruleset to Decision Server like IBM Cloud or on-premise Rule Execution Server.

1. Have Decision Server set up

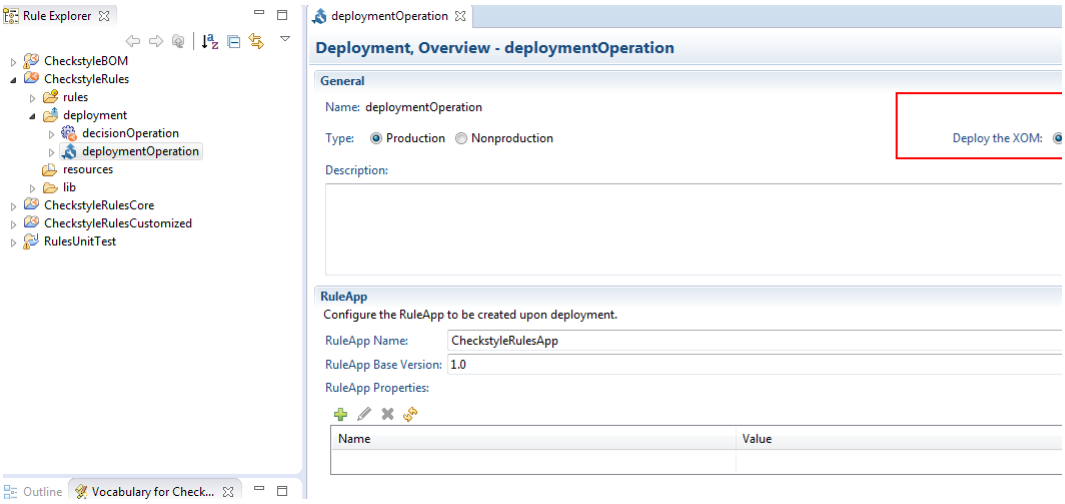
2. Deploy your ruleset

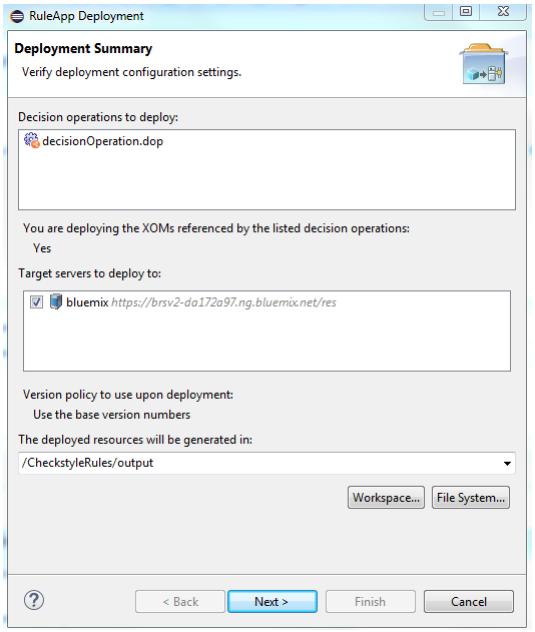
- Open deployment Operation in rule project CheckstyleRules. On the Deployment Target Servers tab, add a target server using the Connection Settings found in Decision Server.

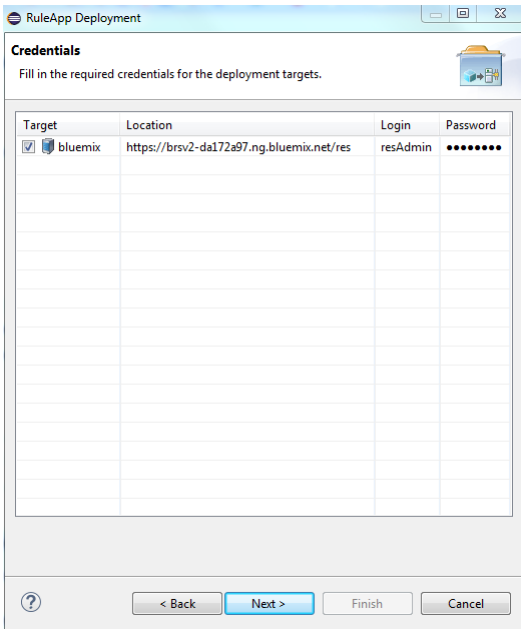


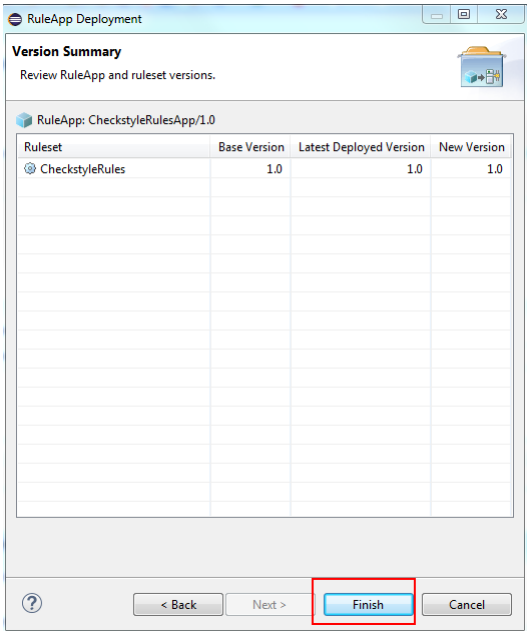


- Deploy the Checkstyle rules to your IBM Business Rules Service on IBM Cloud. Open deploymentOperation in rule project CheckstyleRules. On the Deployment Overview tab, make sure Deploy the XOM is selected as Yes and click Proceed to RuleApp deployment.

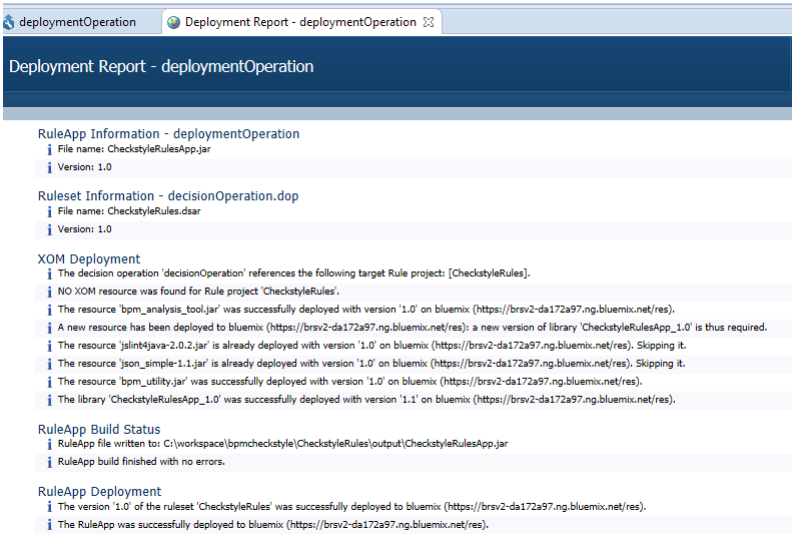

Rule Explorer
CheckstyleBOM
CheckstyleRules
rules
deployment
decisionOperation
deploymentOperation
resources
lib
CheckstyleRulesCore
CheckstyleRulesCustomized
RulesUnitTest
Outline
Vocabulary for Check...
deploymentOperation
Deployment, Overview - deploymentOperation
General
Name: deploymentOperation
Type: Production Nonproduction
Deploy the XOM:
Description:
RuleApp
Configure the RuleApp to be created upon deployment.
RuleApp Name: CheckstyleRulesApp
RuleApp Base Version: 1.0
RuleApp Properties:
Name
Value


RuleApp Deployment
Deployment Summary
Verify deployment configuration settings.
Decision operations to deploy:
decisionOperation.dop
You are deploying the XOMs referenced by the listed decision operations:
Yes
Target servers to deploy to:
bluemix https://brsv2-da172a97.ng.bluemix.net/res
Version policy to use upon deployment:
Use the base version numbers
The deployed resources will be generated in:
/CheckstyleRules/output
Workspace...
File System...
< Back
Next >
Finish
Cancel





- After the deployment completes, a success report is generated as follows. You can also check your IBM Cloud Business Rules service to verify that the ruleset is successfully deployed.



3. Change configurations in ida.properties

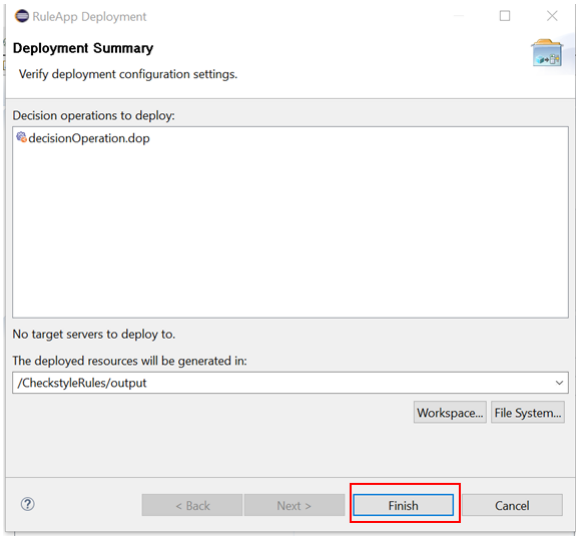
- Make sure the value of ruleExecutionMode in ida.properties is 2.
- Configure the values of resUser, resPassword, resUrl, restUrl and resPort in ida.properties.

Deploy ruleset in local

If you want to execute ruleset in local file system, you need to replace the ruleset file following steps below.

1. Deploy your ruleset

- There is no need to define or choose a target server when deploy your ruleset in local.



- After the successful deployment, the ruleset has been generated in the output folder of CheckstyleRules Project as the file CheckstyleRules.dsar. Replace the ruleset in IDA.war(IDA-web/WEB-INF/classes/data/CheckstyleRules.dsar) with the new one.

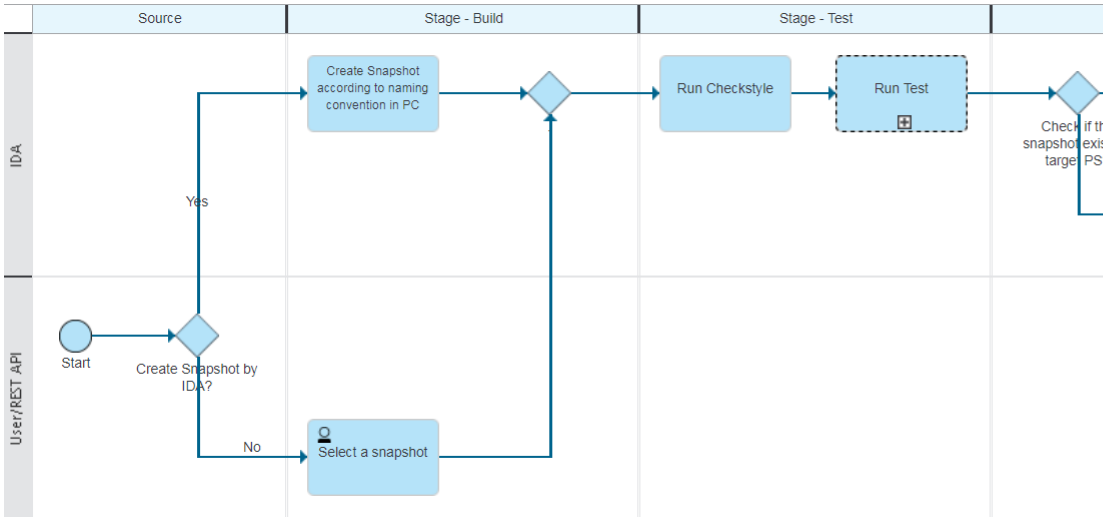
2. Change configurations in ida.properties

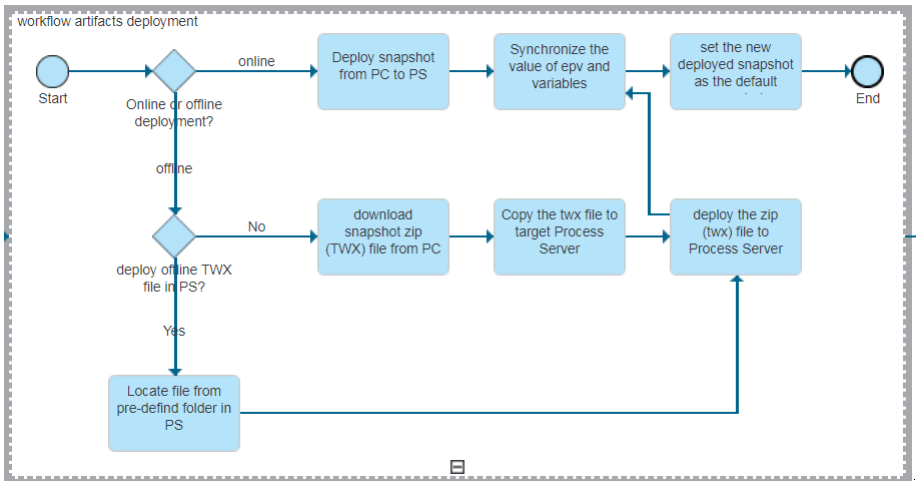
- Make sure the value of ruleExecutionMode in ida.properties is 0.

Pipeline overview

Pipeline overview

Pipelines are workflows that contain individual steps. Each step is responsible for a specific action in the process. The flowchart below is an example of one CI/CD scenario easily modeled in ida pipeline:

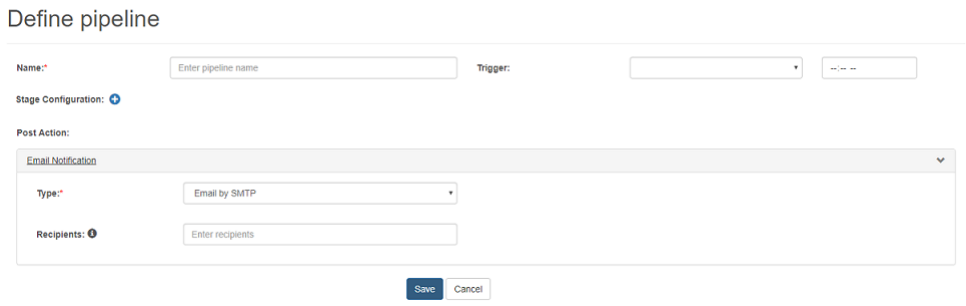




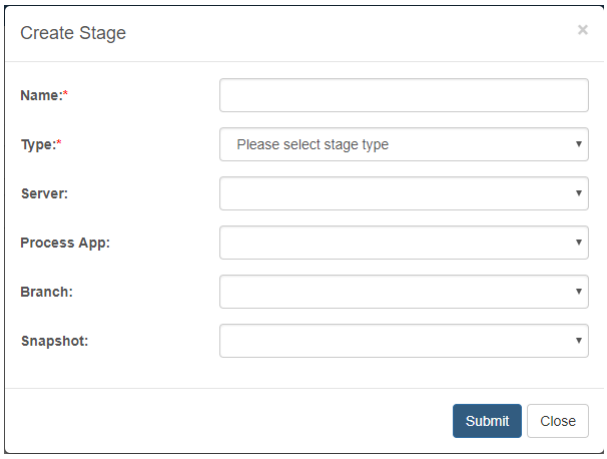
Create a pipeline for continuous deployment

Create a pipeline

1. Click **Create Pipeline** button in pipeline page.



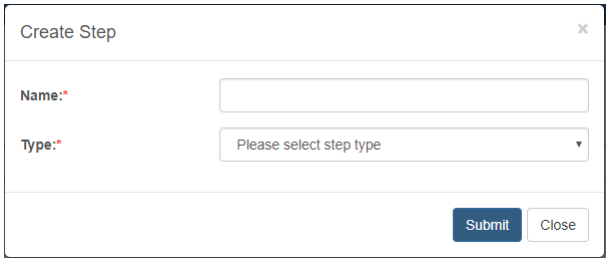
2. Click  icon to add a new stage.



The fields to create a stage is explained as below.

Label	Description
Name	The stage name
Type	The options are Development, Test, Stage, Production and Utility
Server	The server environment based on the stage type, the Utility stage type doesn't have server
Process App	Process apps of the selected server
Branch	The branch of the selected process app
Snapshot	The snapshot of the selected branch

3. Click  icon below the stage step to add a new step in the stage.



The fields to create a step is explained as below.

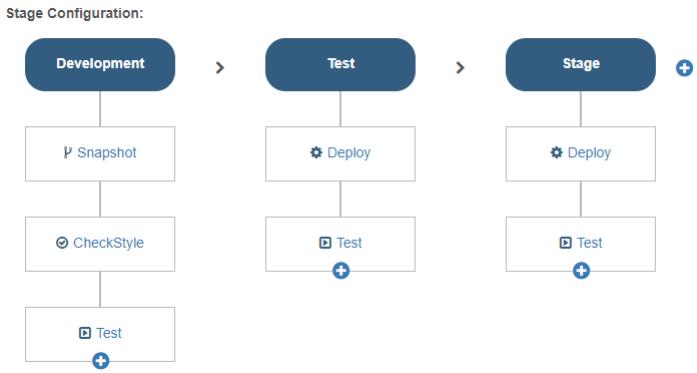
Label	Description
Name	The stage name
Type	The options are Snapshot, Deployment, Checkstyle, Test and Script.

In **Development** type of stage, you can choose the step type as **Snapshot**, **Checkstyle**, **Test** and **Script**.

In **Test**, **Stage** and **Production** type of stages, you can choose the step type as **Deployment**, **Test** and **Script**.

In **Utility** type of stage, you can only choose the step type as **Script**.

4. Click  icon to add more stages, a typical stage looks like below.

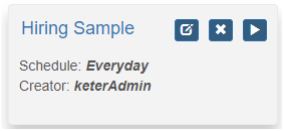


You can't change the sequence of the first stage, but you can adjust the other stages sequence by drag and drop.

You can adjust the steps sequence in a stage by drag and drop.

Edit a pipeline

1. In the pipeline list page, click button.



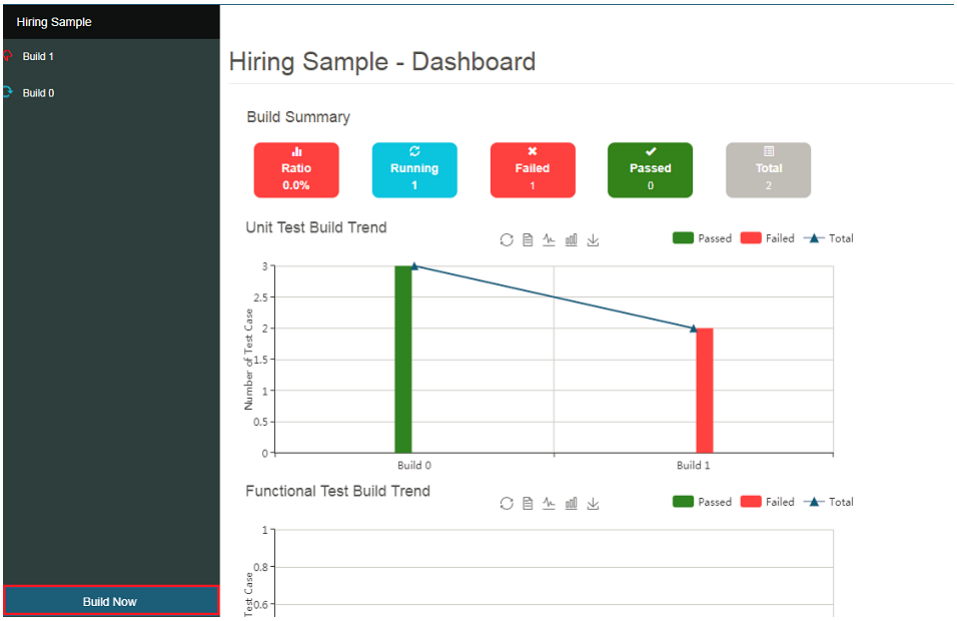
2. Update pipeline form according to the table showed in creating session.

Delete a pipeline

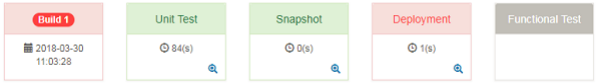
In the pipeline list page, click ✖ button.

Run a pipeline

1. You have two ways to run the pipeline. Firstly, click **Run** button at the top right corner of the pipeline section. Secondly, enter the pipeline dashboard page, click **Build now** button.

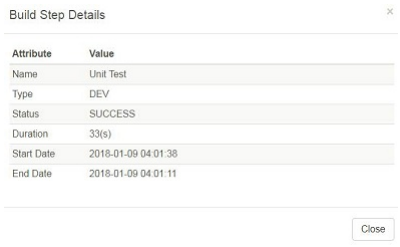


When you run the build, the pipeline build view will appear with every stage building status. Blue color means running, grey means waiting, green means success and red means failure.



Click the build time text in first block, you will see the build logs in popup dialog.

Click icon of the build step, you will see the build step details. It will include build time, duration etc information.



2. When the pipeline build is over, the summary information will appear in pipeline dashboard. The step which is failed will not run and there is not report. You can click the **build no** link to see the whole report. Details of the report

Deploy snapshot from PC to PS

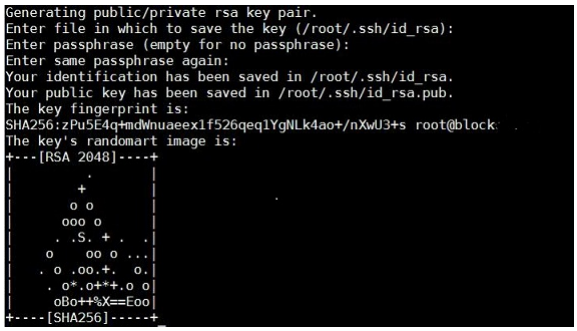
IDA pipeline allows you to deploy your snapshot from PC to PS, either through online or offline deployment. Testers can easily test projects on PS side. To deploy snapshot from PC to PS by IDA, there are some configurations needed. In below instruction, we assume the BPM Process Center IP is 192.168.0.10.

Create SSH Key

1. Login into the IDA server and create the key pair in IDA server.

   ```
   $ssh-keygen -t rsa
   ```

2. After key generation process, you can see below output.



   The public key is now located in /root/.ssh/id_rsa.pub. The private key (identification) is now located in /root/.ssh/id_rsa.

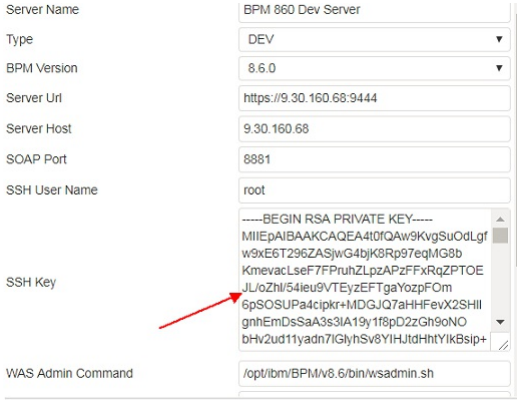
3. Copy the public key to the PC Server and PS Server. Copy id_rsa.pub file to BPM Server (PC/PS) /tmp folder, and execute below command.

   ```
   cat /tmp/id_rsa.pub >> ~/.ssh/authorized_keys
   ```

   Notes:

The Linux operator user should have the permission to run “wsadmin.sh”

4. Use cat ~/.ssh/id_rsa to print your private key. Copy this key content to your BPM PC and PS configuration in IDA. Also you need to fill the WAS admin command path.



5. In PS Server configuraion, you need to fill the connect server name.

Verify SSH Key

1. If you are in Linux environment, copy the id_rsa to BPM server /tmp folder. You can use below command to vertify the connection. Please replace your IP address accordingly.

```
chmod 600 /tmp/id_rsa
ssh -i /tmp/id_rsa root@192.168.0.10
```

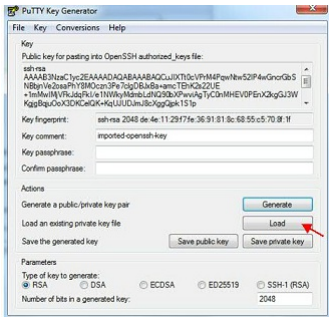
2. If you are in Windows environment, you can use Putty tool to vertify it.

   Putty: https://the.earth.li/~sgtatham/putty/latest/w32/putty.exe
   Puttygen: https://the.earth.li/~sgtatham/putty/latest/w32/puttygen.exe

   a. Open puttygen.exe

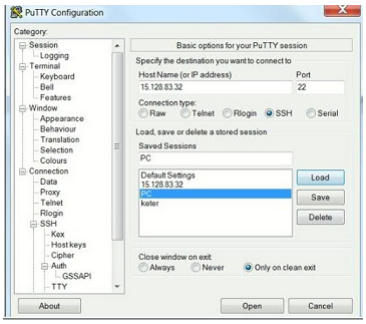
   b. Open Conversions -> Import Key

   c. Select id_rsa file

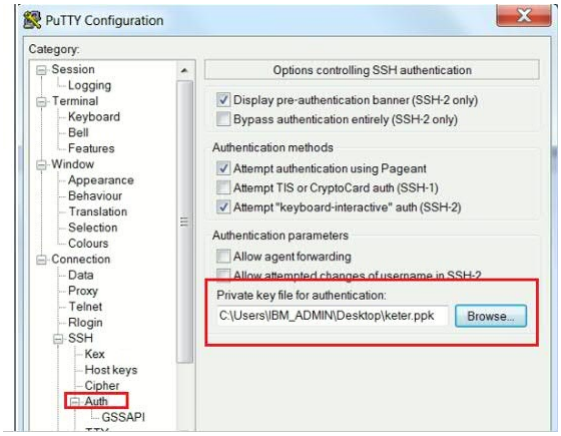


   d. Click Save private key button, save as IDA.ppk .

e. Open putty.exe, For example enter PC IP address 192.168.0.10



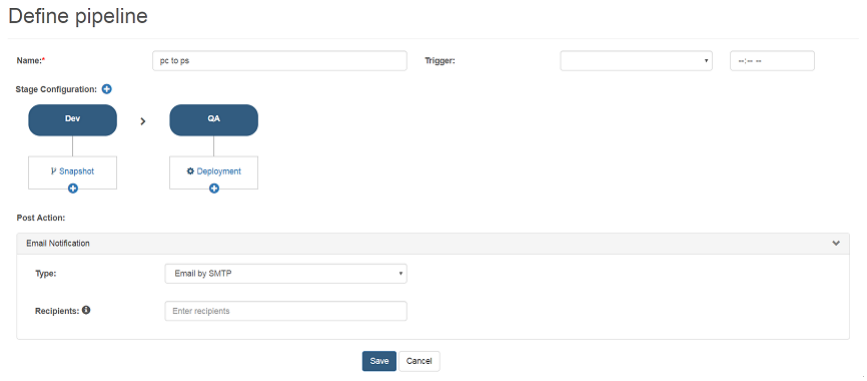
f. Select category: Connection -> SSH -> Auth, and set the private key file, then click Open button.



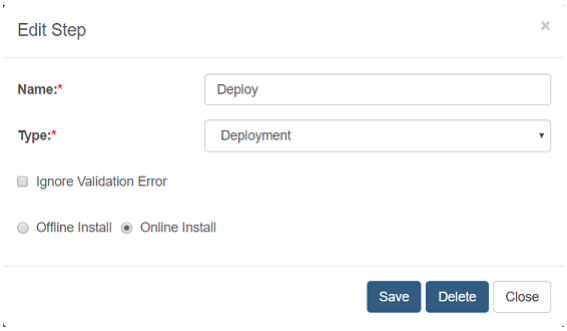
g. You can find you can use ssh to login into that server.

Define pipeline from PC to PS through online deployment

1. Define the pipeline for deployment snapshot from PC to an online PS.



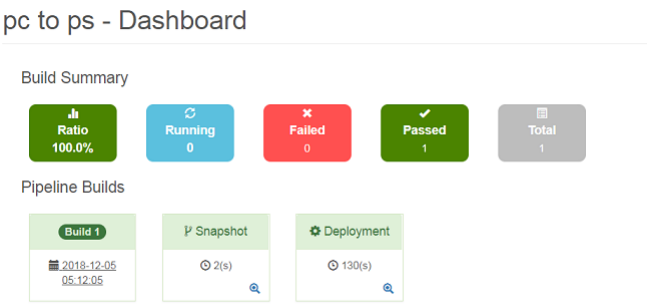
2. For the Deloyment step, you need to select **Online Install** for it.



3. Run the pipeline, you will be prompted with a popup window to provide the snapshot acronym, which is a optional field. If the snapshot acronym is empty, the snapshot created in DEV (PC) stage will be installed to online PS. If the snapshot acronym is provided, it means the existing snapshot in DEV (PC) will be located

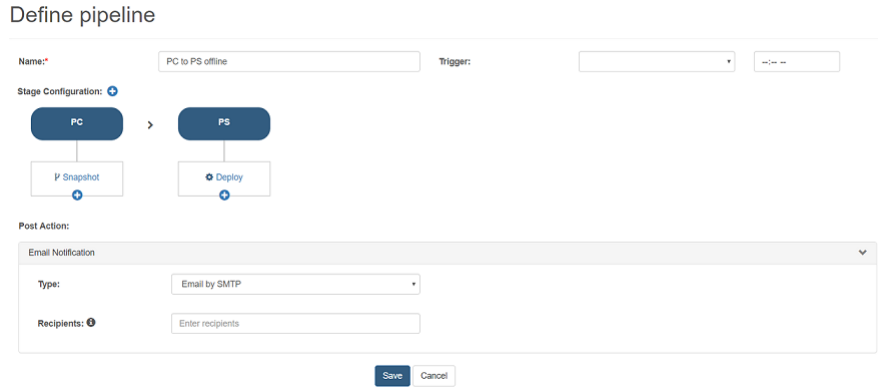
by snapshot acronym and installed to online PS.

4. After the pipeline is started, then you can check the pipeline build status. If sucessful, it means the snapshot is deployed from PC to online PS.

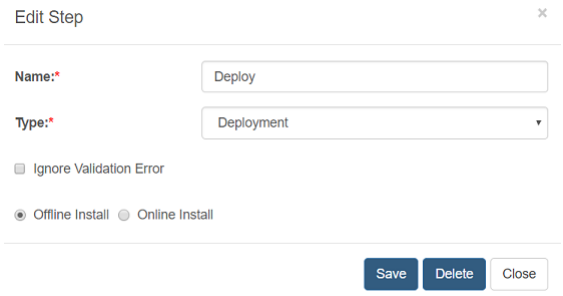


Define pipeline from PC to PS through offline deployment

1. Define the pipeline for deployment snapshot from PC to an offline PS.

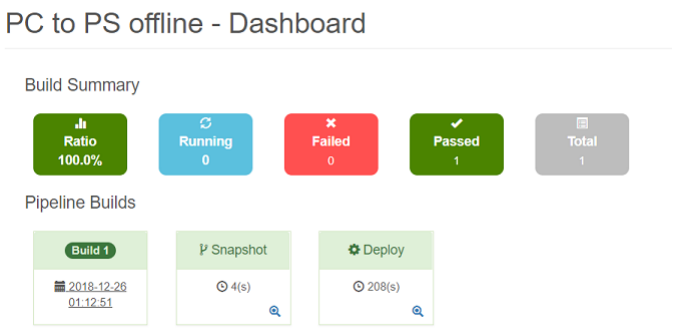


2. For the Deloyment step, you need to select **Offline Install** for it.



3. Run the pipeline, you will be prompted with a popup window to provide the snapshot acronym, which is a optional field. If the snapshot acronym is empty, the snapshot created in DEV (PC) stage will be installed to offline PS. If the snapshot acronym is provided, it means the existing snapshot in DEV (PC) will be located by snapshot acronym and installed to offline PS.

4. After the pipeline is started, then you can check the pipeline build status. If sucessful, it means the snapshot is deployed from PC to offline PS.



Known Limitions:

IDA machine, PC Server, PS server should be Linux server.

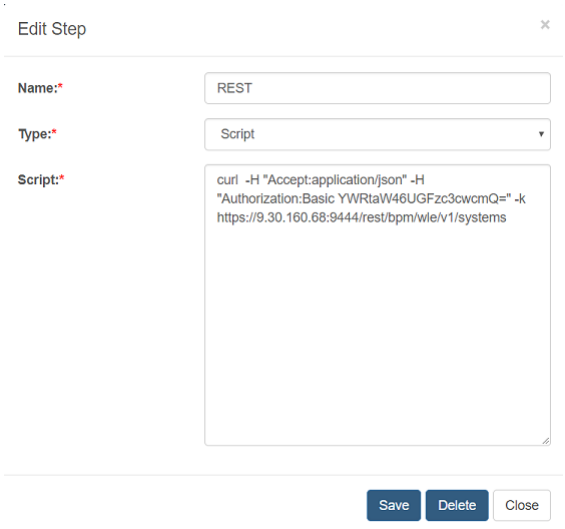
Pipeline script

Pipeline script summary

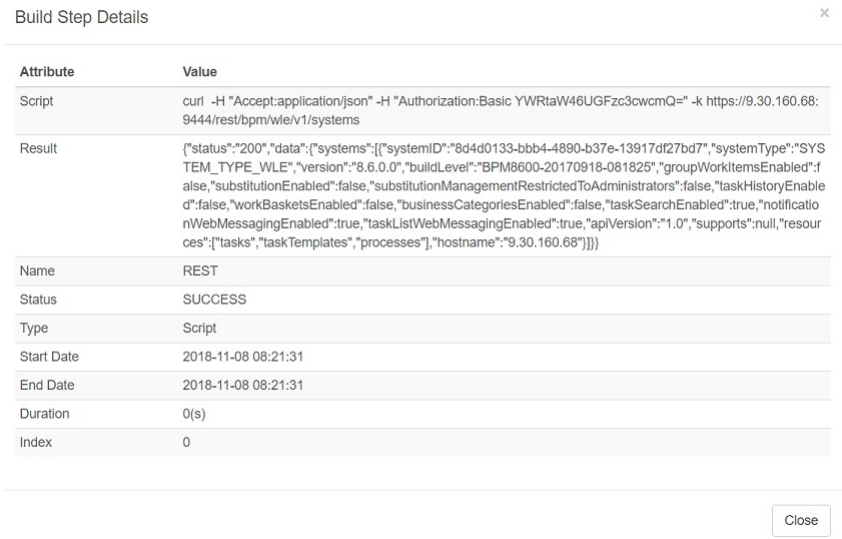
In pipeline stage step definition, there is a step called script, which allows users to execute script in the server where IDA is deployed.

Define script

In **Edit Step** modal, select **Script** as **Type** then you can define one or more scripts in **Script** text area. For multiple scripts, each of them need to start from a new line.



After the pipeline is executed, you can view the script execution result.



Script supported parameters

IDA supports below parameters in Script. They can be used in Script to represent Pipeline related attributes.

PIPELINE_NAME * * : ***currentpipelinename*** < ***br*** > * ***{PIPELINE_ID}**: current pipeline ID
STAGE_NAME * * : ***currentstagename*** < ***br*** > * ***{STEP_NAME}**: current step name
BUILD_ID * * : ***currentbuildid*** < ***br*** > * ***{BUILD_REPORT_URL}**: current build report URL
APP_ACRONYM * * : ***currentprocessAppacronymname*** < ***br*** > * ***{SNAPSHOT_ACRONYM}**: current snapshot acronym name

Script samples

Call RESTFul service

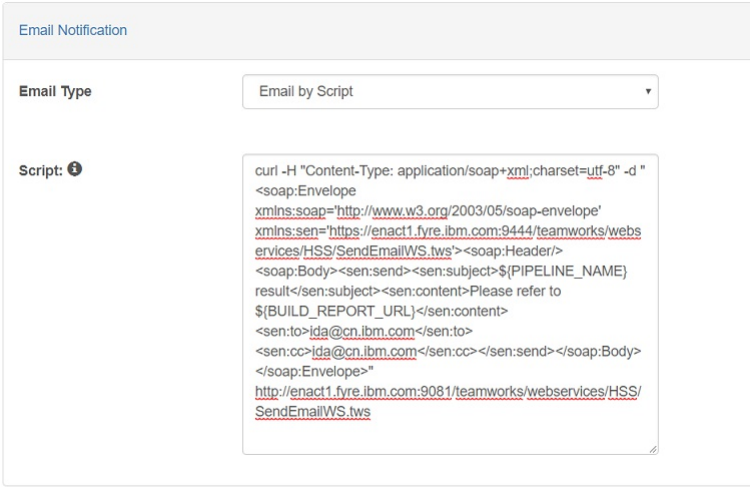
You can use **curl** to call a RESTful service or Web Service in Script. For example, below script calls a BPM REST API by curl.

curl -H "Accept:application/json" -H "Authorization:Basic YWRtaW46UGFzc3cwcmQ=" -k https://9.30.160.68:9444/rest/bpm/wle/v1/systems

Call Web Service

You can also use **curl** to call a Web Service. For example, you can call a Web Service to send email notification during pipeline creation. Assume the Web Service is based on SOAP 1.2 and its WSDL URL is: http://enact1.fyre.ibm.com:9081/teamworks/webservices/HSS/SendEmailWS.tws?wsdl, you can use below curl script to call it.

curl -H "Content-Type: application/soap+xml;charset=utf-8" -d "${PIPELINE_NAME} resultPlease refer to ${BUILD_REPORT_URL}ida@cn.ibm.comida@cn.ibm.com" http://enact1.fyre.ibm.com:9081/teamworks/webservices/HSS/SendEmailWS.tws



Call wsadmin command

You can execute a wsadmin command in Script. The wsadmin command is running against the BPM server associated to the Stage BPM configuration. For example,

ssh AdminTask.BPMSetEnvironmentVariable('[-containerAcronym ${APP_ACRONYM} -containerSnapshotAcronym ${SNAPSHOT_ACRONYM} -environmentVariableName TEST_KEY -environmentVariableValue 8899]')

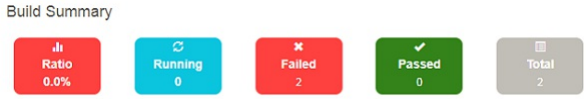
This Script first logon BPM server using ssh, then execute the wsadmin commmand there to update the BPM environment variable. The format of the Script to call wsadmin command is

ssh + space + **wsadmin command**

Pipeline report

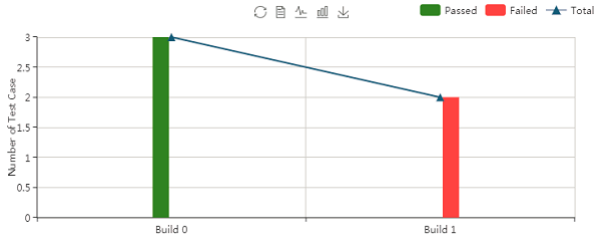
Pipeline build summary

In pipeline dashboard, you can see the pipeline build summary. **Ratio** is the success rate of pipeline build, **Running** is the number of running pipeline build, **Failed** is the number of failed pipeline build, **Passed** is the number of successful pipeline build and **Total** means the number of total pipeline build.



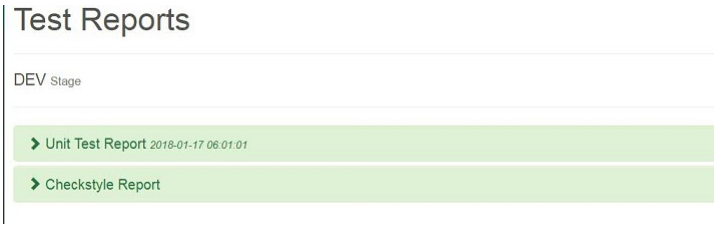
Test build trend

In pipeline dashboard, you can see three kind of test build trend: **Unit Test Build Trend**, **Functional Test Build Trend(QA)** and **Staging Functional Test Build Trend**. It supports conversion to data view, line chart and pie chart. You can also download the image by clicking **SaveASImage** icon in tool bar.



Pipeline report

Every successful step will have a report in **Test Reports** page. You can read the report you want by clicking links.



Click the **Unit Test Report** link,you can read the whole unit testing report of this project. It includes test summary statistic of every test suite, test case, test step.

Click the **checkstyle report** link to see the result. The same way to read other reports.

Pipeline stage & step reference

Pipeline stage reference

Pipeline stage is like a milestone which contains one or more pipeline steps. Each stage has a **Type** attribute, which impacts the supported pipeline steps. There are 3 types of stage, which is listed as below.

Stage Type	Supported Steps	Description
Development	Snapshot, Checkstyle, Test, Script	The stage is associated to BPM PC, where users can run Snapshot, Checkstyle, Test, Script. When you define the **Development** stage, you need to select the BPM server, ProcessApp, and Snapshot. The snapshot can be overridden if there is a **Snapshot** step in this Stage, or users specify another snapshot when running the pipeline. If is not overridden, this snapshot will be used for Checkstyle, Test and Deployment.
Test, Stage, Production	Checkstyle, Test, Script	The stage is associated to BPM PS, where users can run Checkstyle, Test, Script. Please note that the Snapshot step can only be run from Development stage. If it is the first stage in pipeline, it means there is no associated PC environment thus no new snapshot will be created during the pipeline execution. So when you run the pipeline, you need to specify an existing snapshot name to execute Checkstyle, Test, and Deployment steps.
Utility	Script	The stage is not associated to BPM environment. It is associated to IDA server, which you can run Script.

Pipeline step reference

Step Name	Parameters	Description
Snapshot	**Naming Pattern**: the naming pattern for the created snapshot.	The Snapshot step is used to create a snpshot by specified naming pattern. This step is only available in BPM DEV environment.
Checkstyle	**Health Score Threshold**: the threshold to define the health score. If **Halt On Failure** is checked and the health score exceeds the threshold, the checkstyle step is considered to be failed and the pipeline is stopped. **Warnings Threshold**: the threshold for the number of warnings. If **Halt On Failure** is checked and the number of warnings exceeds the threshold, the checkstyle step is considered to be failed and the pipeline is stopped. **Ignore Empty Documentation**: indicate whether to ignore empty documentation. **Ignore JS Coding Style Check**: indicate whether to ignore JS coding style check. **Halt On Failure**: indicate whether to stop the pipeline if the checkstyle result exceeds the threshold. **Tip**: indicate whether to run the checkstyle against TIP snapshot.	The Checkstyle step is used to run the checkstyle against a processApp.
Test	**Project**: the test project to run. **Tip**: indicate whether to run the test against TIP snapshot. **Exclude test cases**: indicate whether to exclude test cases. Once checked, all the test cases in the test project will be listed. User can select one or more test cases that will be excluded from the test execution.	The Test step is used to run the testing project.
Deployment	**Ignore Validation Error**: indicate whether to ignore validation errors during deployment. **Offline Install** : indicates it is an offline deployment. **Online Install** : indicates it is an online deployment.	The Deployment step is used to deploy the processApp snapshot to process server, either through onine or offline deployment. The step is only available in BPM non-DEV environment(QA, STG, PROD).
Script	**Script**: the script content to execute	The Script step is used to run a script in IDA server. Refer to this for more details about pipeline Script samples.

Monitor Management

About this task

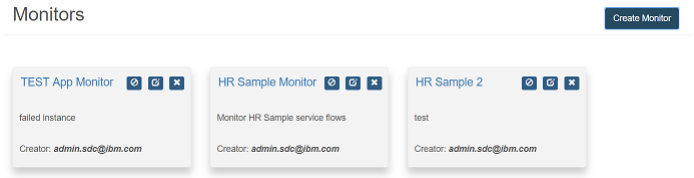
In this task, you will create a new BPM monitor, edit monitor, delete monitor, disable/enable monitor and view monitor charts.

Prerequisite

To use Monitor feature, target BPM system to be monitored has to integrate with IDA through DEF. For details, please refer toConfigure BPM DEF.

Create monitor

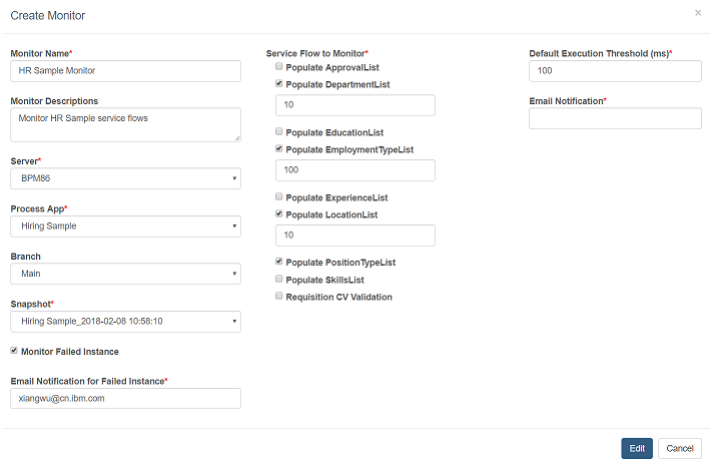
1. In the monitor list page, click **Create Monitor** button.



2. Fill the monitor form according to the following table.

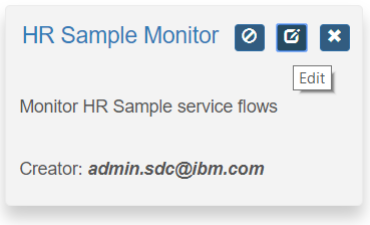
Parameter	Description
Monitor Name	Name of monitor
Monitor Descriptions	Descriptions of monitor
Server	Server environment
Process App	Process apps of the selected server
Branch	The branch of selected process app or toolkit
Snapshot	The snapshot of selected branch
Monitor Failed Instance	Indicate whether to monitor failed instance
Email Notification for Failed Instance	Email address to receive notification when instance is failed. If multiple email addresses are needed, separate them by comma.
Service Flow to Monitor	Select service flows to be monitored
Execution Threshold (ms)	Define execution threshold in millisecond for each selected service flow
Default Execution Threshold (ms)	Define the default execution threshold in millisecond for all service flows as a global value in this module. For each selected service flow to be monitored, if the Execute Threshold (ms) is not specified, the Default Execution Threshold (ms) is used. Otherwise, the Execute Threshold (ms) overwrites the Default Execution Threshold (ms)
Email Notification	Email address to receive notification when service flow actual execution time exceeds the threshold. If multiple email addresses are needed, separate them by comma.

3. Click **Create** button, you will return to the monitor list page and find the new one there.



Edit monitor

1. Click **Edit** icon for the monitor you want to edit in the monitor list page.



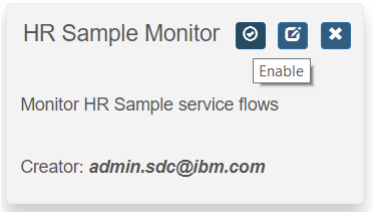
2. Update monitor form according to the table showed in creating section.

Disable monitor

1. Click **Disable** icon for the monitor you want to disable in the monitor list page.

Enable monitor

1. Click **Enable** icon for the monitor you want to enable in the monitor list page.

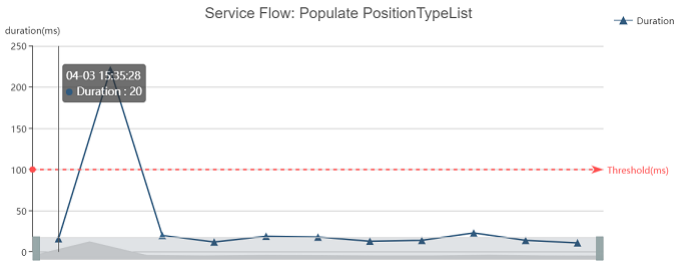


Delete monitor

1. Click **Remove** icon for the monitor you want to delete in the monitor list page.

View monitor chart

1. Click monitor name for the monitor you want to view chart in the monitor list page. Then you will drill down to the service flow performance line chart.



Supporting Tool

About this task

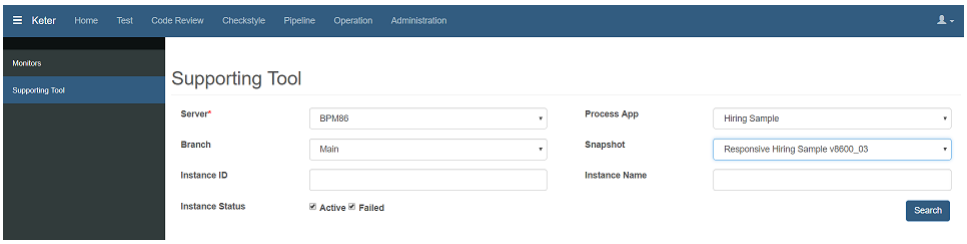
In this task, you will use the BPM supporting tools to search instance, migrate instance between snapshots, update task data, finish task, move token, and re-try failed instance.

Search instance

1. In **Operation page**, click **Supporting Tool** menu.
2. Fill the instance search form according to the following table.

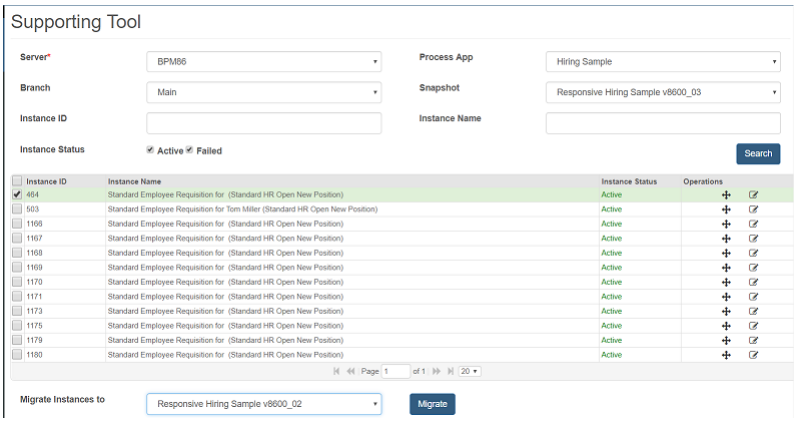
Parameter	Description
Server	Server environment
Process App	Process apps of the selected server
Branch	The branch of selected process app or toolkit
Snapshot	The snapshot of selected branch
Instance ID	ID of instance
Instance Name	Name of instance
Instance Status	Status of instance. Must be Active or Failed or both.

3. Click **Search** button, you will see the instance table as search result.



Migrate instance

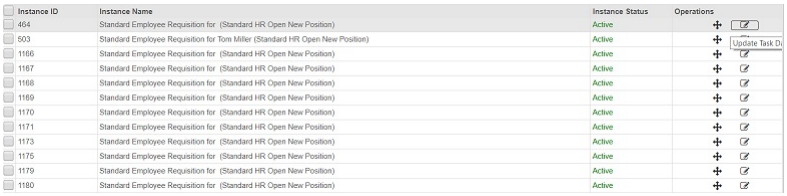
1. Select one or more instances in the search result table, choose a target snapshot from **Migrate Instance to** drop down list, the click **Migrate** button.



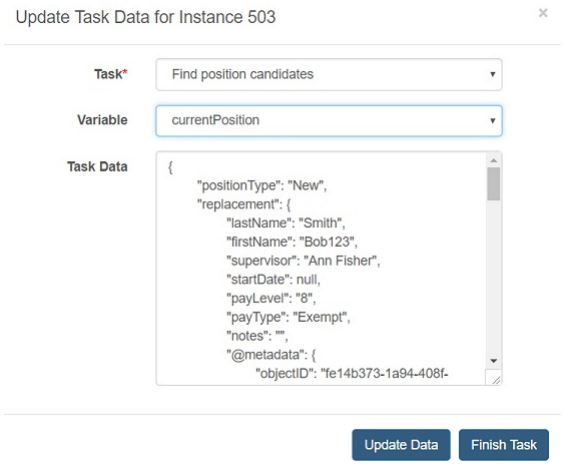
2. As the result, the instance is migrated from source snapshot (snapshot selected in the search form) to target snapshot (snapshot selected in the **Migrate Instance to** drop down list).

Update task data

1. Locate the instance row then click **Update Task Data** icon in **Operations** column.



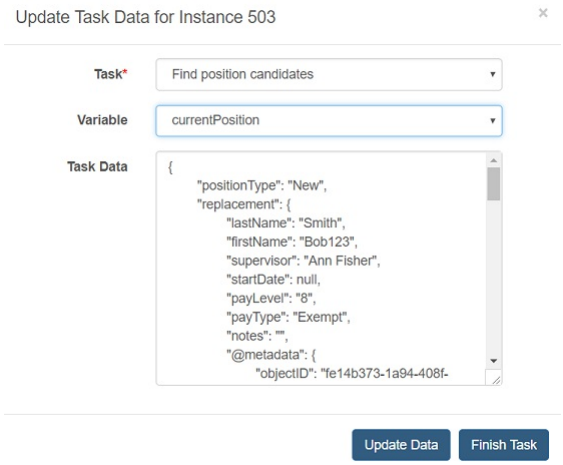
2. In the popup window, select **Task** name. The task information is retrieved from the instance details.
3. Select **Variable** name. The variable information is retrieved from the task selected above.
4. Once the **Variable** is selected, the **Task Data** is retrieved. You can update the task data then click **Update Data** button to update the task data.



Finish task

1. Locate the instance row then click **Update Task Data** icon in **Operations** column.

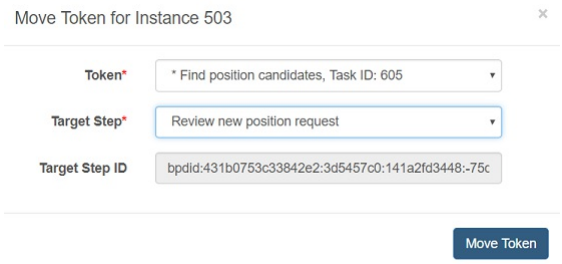
2. In the popup window, select **Task** name. The task information is retrieved from the instance details.
3. Select **Variable** name. The variable information is retrieved from the task selected above.
4. Once the **Variable** is selected, the **Task Data** is retrieved. You can update the task data then click **Finish Task** button to update the task data and finish task.



Move token

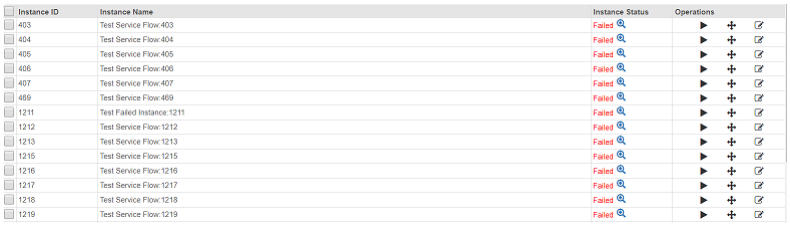
1. Locate the instance row then click **Move Token** icon in **Operations** column.

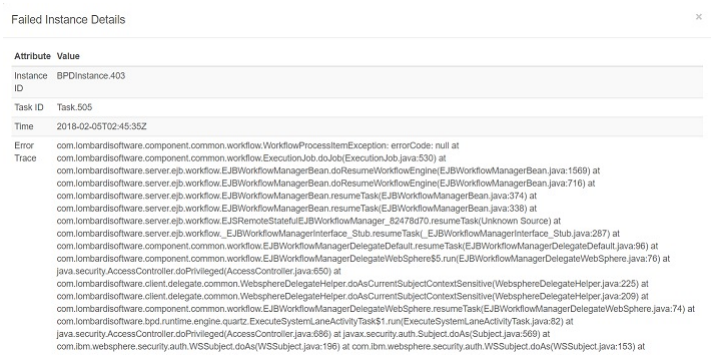
2. In the popup window, select **Token** name. The token information is retrieved from the instance details.
3. Select **Target Step** name then click **Move Token** button to move the instance token to target step.



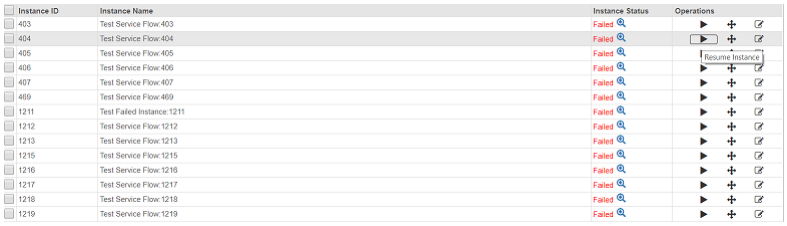
Retry failed instance

1. For failed instance, you can click the view icon in **Instance Status** column to view the detailed error trace.





2. Failed instance can be re-tried by clicking the **Resume Instance** icon in **Operations** column.



3. As the result, the **Instance Status** will be updated (Active or Failed).

Users management

Add a new user

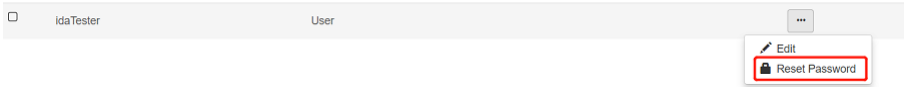
Use super account (idaAdmin/idaAdmin) login into IDA application. 1. Click **Administrator** tab, then switch to **Users management** tab to add user.

2. Click 

icon which is on the right of **Users Management** to add new users. 3. Fill out the form below. Then click **Create** button.

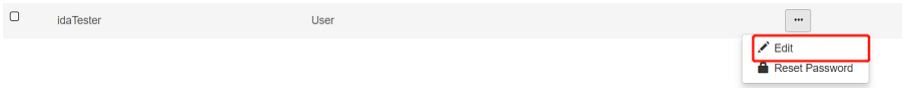
```
![][administrator_add]

| Field | Description |
| --- | --- |
| User Name | Use email address as user name|
| Role | A user can have multiple roles, only Admin role user can configure settings|

Permissions for each role are shown in the following figure:

![][administrator_role_permission]

**Notes:**

Role and role privileges can be customized according to client needs.
```

4. Click **Reset password** button can reset password of one user. The default password is **123456**.



Edit a user

1. Click **Edit** icon on the right of the user that you want to edit.



2. Change roles of the user. Then click **Save**.

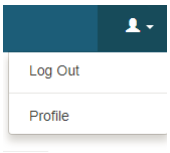
Activate/Deactivate users

Select users in the **Users Management** table, then click **Activate** button to activate the selected inactive users. And click **Deactivate** button to deactivate the selected active users.

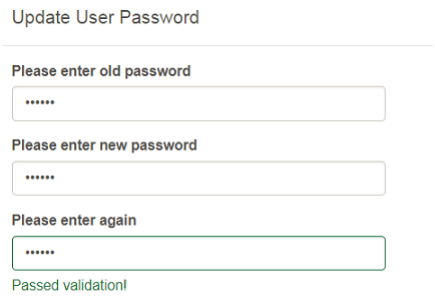
```
![][administrator_active_user]
```

User profile

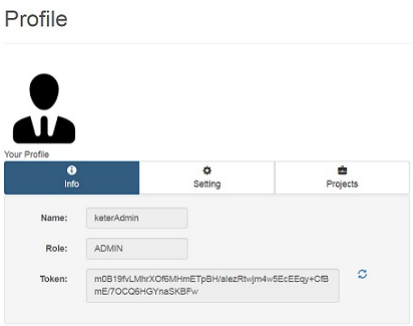
1. On the top right corner, click the **Profile** icon.



2. Click **Info** button to get basic information of your account.
3. Click **Setting** button to change your password.



4. Click **Projects** button to get basic information of your projects.

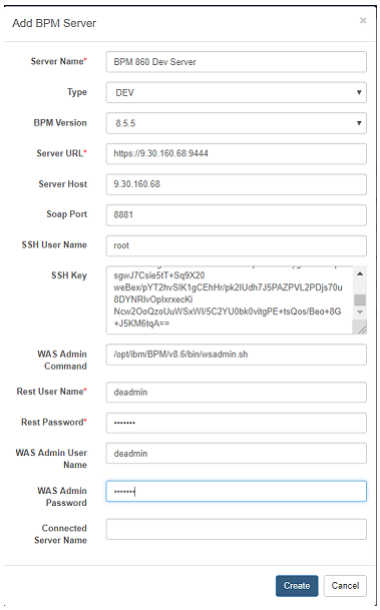


BPM configuration

Add a BPM server

1. Click **Administrator** tab, then switch to **BPM Configurtaion** tab to manage BPM server.
2. Click 

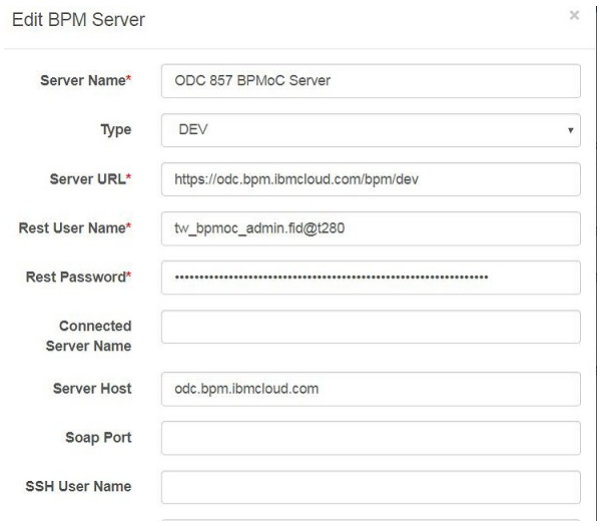
icon which is on the right of **BPM Server Configuration** to add new BPM server.
3. Fill out the form below. Then click **Create** button.



Field	Description
Server Name	the BPM server name
Type	DEV, QA, STG, PRO
BPM Version	the BPM server version
Server Url	the BPM server URL, eg: https://bpmserver:9443
SOAP port	soap port defined in Was console
SSH User Name	used for snapshot deployment from PC to PS
SSH key	used for snapshot deployment from PC to PS
WAS admin Command	the path of wsadmin.sh in linux server
Rest User Name	BPM server rest api username
Rest Password	BPM server rest api password
WAS Admin username	WAS admin name
WAS Admin password	WAS admin password
Connected Server Name	online PS server name

Notes:

If you need to add bpmoc server,you don't need to fill everything only add bpm server url,server type,hostname and rest username(function id) and password.
If you login get 401 unauthorized error,please check your function id belong to tw_admins (administrative access) and tw_authors (non-administrative access) groups.



Edit BPM server

1. Click **Edit** icon on the right of the BPM server that you want to edit.

2. Change configurations of the BPM server. Then click **Save**.

Delete BPM server

Select the servers in the **BPM Configuration** table, then click the **delete** icon to delete BPM server.

Test BPM server

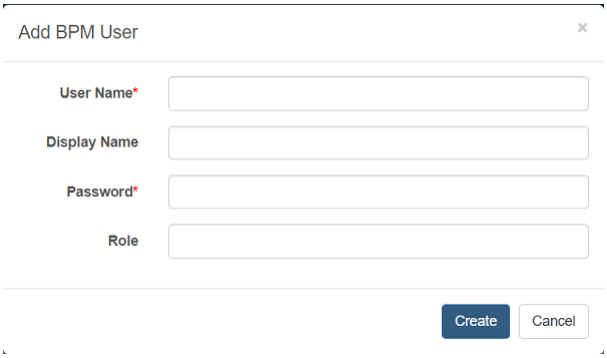
Select **Test Connection** icon on the right of the BPM server that you want to test.If test connection is failed,it will popup warning message.

Add user to a BPM server

1. Click **Edit BPM User** icon on the right of the BPM server. Then you can see then BPM user list of the server.

2. Click 

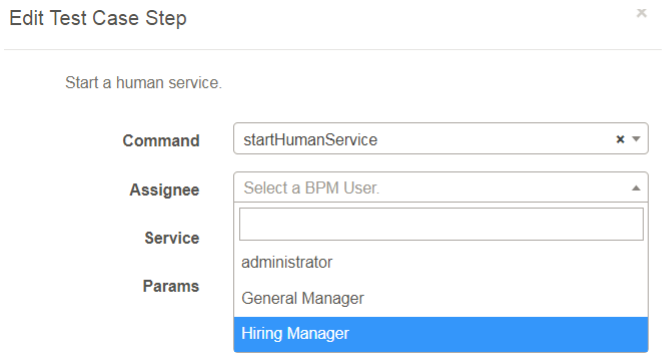
icon which is on the right of **BPM User Configuration**. Then fill out the form below to add a new user for the server.



Field	Description
User Name	the user name of BPM server
Display Name	the display name of BPM user
User Password	the password of BPM user
Role	the logical role name of BPM user

After adding user, you can see the user list for this BPM server.

Notes:
Pls make sure the added user exist in your selected BPM server.Also You need to well define the **Display name** of the BPM user.You can choose the **Assignee** display name to choose user for the test case step command.



What's more, you can also edit a bpm user in a similar way to editing a user.

Selenium Hub configuration

How to configure selenium hub

You need to setup the Post Installation Setup before this step. ### Add selenium hub 1. Click **Administrator** tab, then switch to **Selemium Hub Configuration** table to manage hubs.

2. Click 

icon which is on the right of **Users manager** to add new users. 3. Fill out the form below. Then click **Create** button.

```
![][administrator_hub]

  Field               Description
 -------------------  ---------------------------
 Server Name          Hub name
 Server URL           Hub URL name
 Browser              Support Firefox, Chrome and IE
 Configuration        Customize browser options for selenium
```

Notes: You can customize the selenium configuration based on the template.Currenctly we support below types of configuration.

Headless for chrome and firefox

```
arguments:
## Enable headless mode
- "--headless"
```

experimentalOption for chrome setting

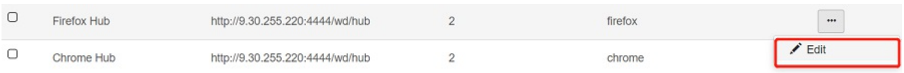
```
experimentalOptions:
## Fix issue "Loading of unpacked extensions is disabled by administrator."
  useAutomationExtension: false
```

Window Size

```
arguments:
## Change windows szie
- "window-size=1920,1080"
```

Edit selenium hub

1. Click **Edit** icon on the right of the Selenium Grid Configuration that you want to edit.



2. Change configurations of the selenium grid. Then click **Save**.

Delete selenium hub

Select the hub in the **Selemium Hub Configuration** table, then click the **Delete** button to delete selenium hub.

Test selenium hub

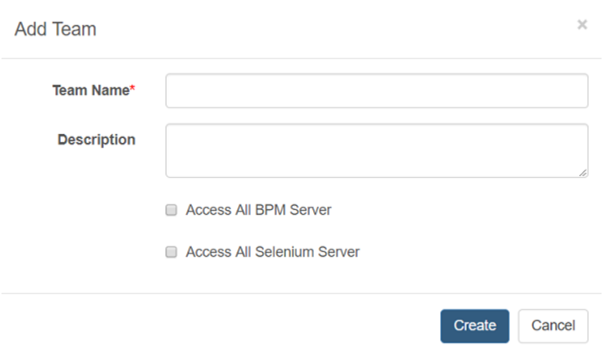
Click **Test Connection** icon on the right of the Selenium Grid Configuration that you want to test.If connection is wrong,it will popup warning info.

Team configuration

Add a team

1. Click **Administrator** tab, then switch to **Team Configurtaion** tab to manage team.
2. Click 

icon which is on the right of **Team Management**.
3. Fill out the form below. Then click **Create** button.



Field	Description
Team Name	the name of the created team

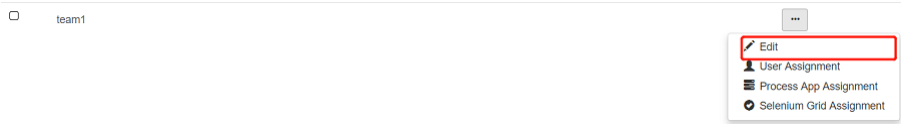
Field	Description
Description	more description or information of the created team
Access All BPM Server	If check, then the users in this team can access all bpm serers without assignment.
Access All Selenium Server	If check, then the users in this team can access all selenium serers without assignment.

Notes:

If checkboxes *Access All BPM Server* and *Access All Selenium Server* are checked, then the team is an administration team. Its users can access all bpm servers, process apps and selenium grids, which does not have to assign. *Process App Assignment* and *Selenium Grid Assignment* buttons are not available.

Edit a team

1. Click **Edit** icon on the right of the team that you want to edit.



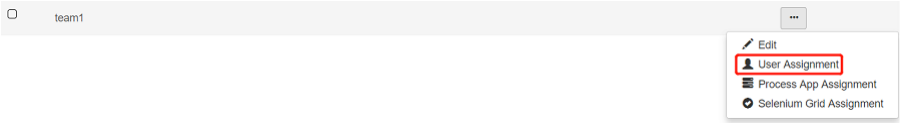
2. Change configurations of the team. Then click **Save**.

Delete teams

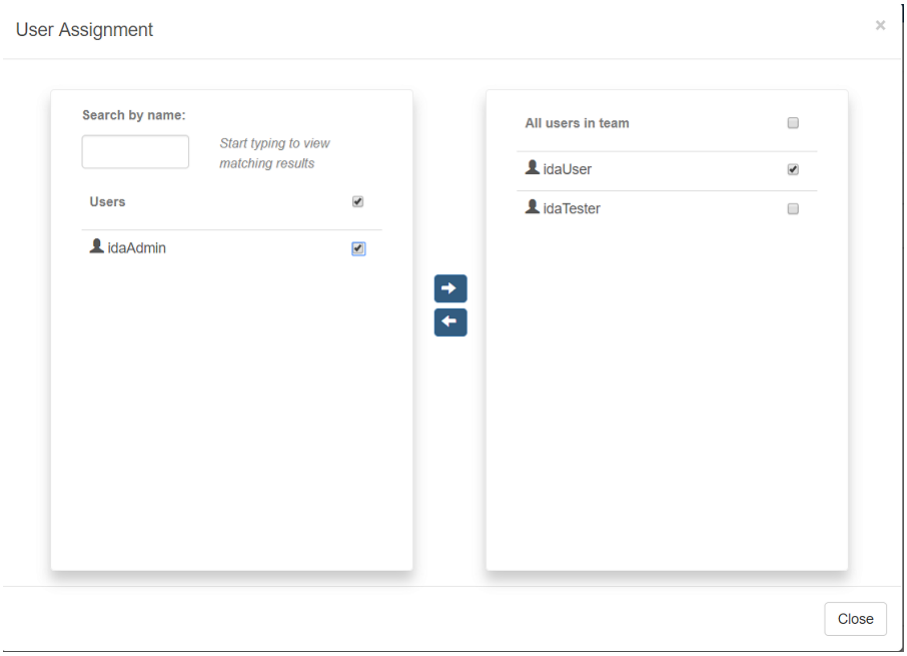
Select the teams in the **Teams Management** table, then click the **delete** icon to delete teams.

Assign users to a team/ Remove users from a team

1. Click **User Assignment** icon on the right of the team that you want to assign or remove users.

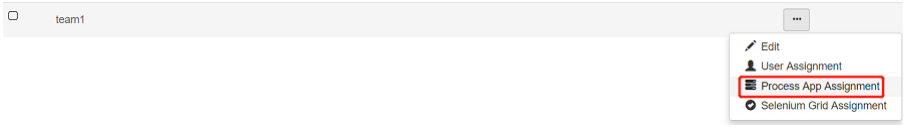


2. The left table of modal lists the users who do not belong to team, while the right do. And you can select users in the left table and click to assign users to the team. Similarly, you can select uses in the right table and click to remove users from the team.

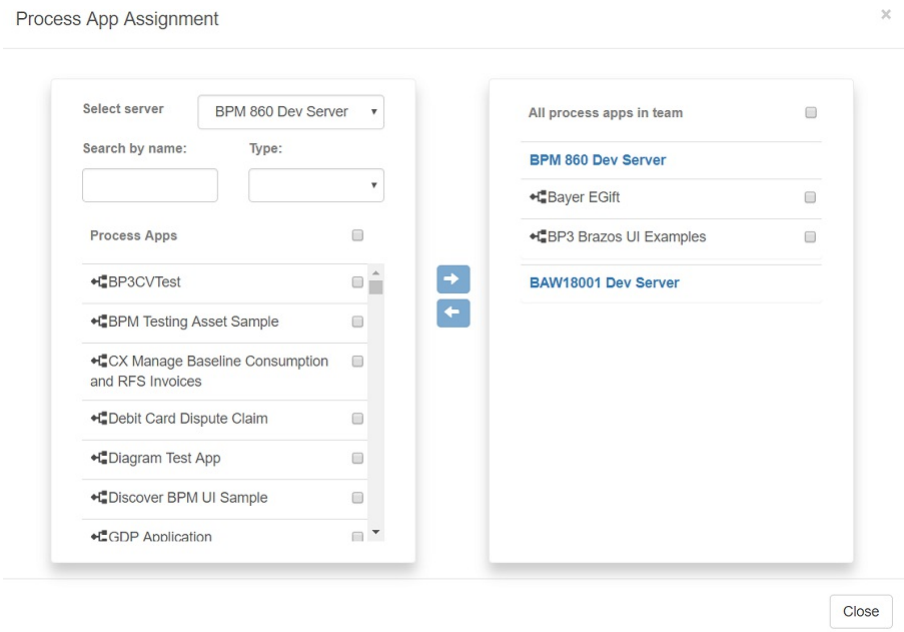


Assign process apps to a team

1. Click **Process App Assignment** icon on the right of the team that you want to assign or remove process apps.



2. Select server and the left table of modal lists the process apps which belong to the server but do not belong to the team.
3. Select process apps in the left table and click  to assign process apps to the team.
4. After finishing step 3, the assigned process apps will appear below the selected server.

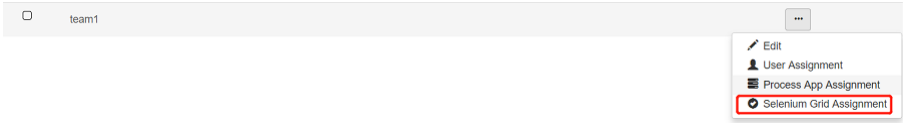


Remove process apps from a team

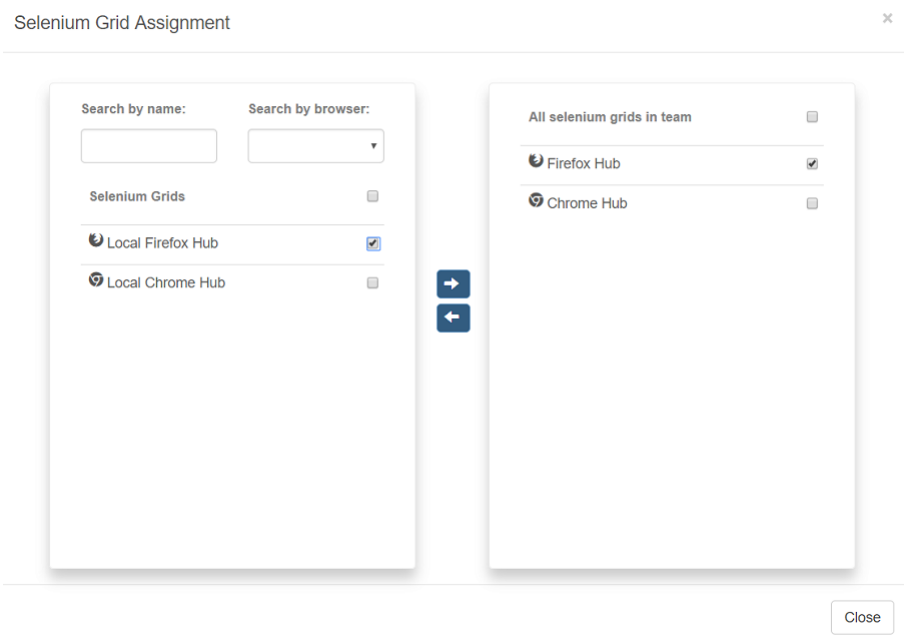
1. Click server and the right table of modal lists the process apps which are under the server and belong to the team.
2. Select process apps in the right table and click  to remove process apps from the team.
3. After finishing step 2, the selected server in step 1 will be selected in the left modal and the removed process apps will in the lists which belong to the server but do not belong to the team.

Assign selenium grids to a team/ Remove selenium grids from a team

1. Click **Selenium Grid Assignment** icon on the right of the team that you want to assign or remove selenium grids.



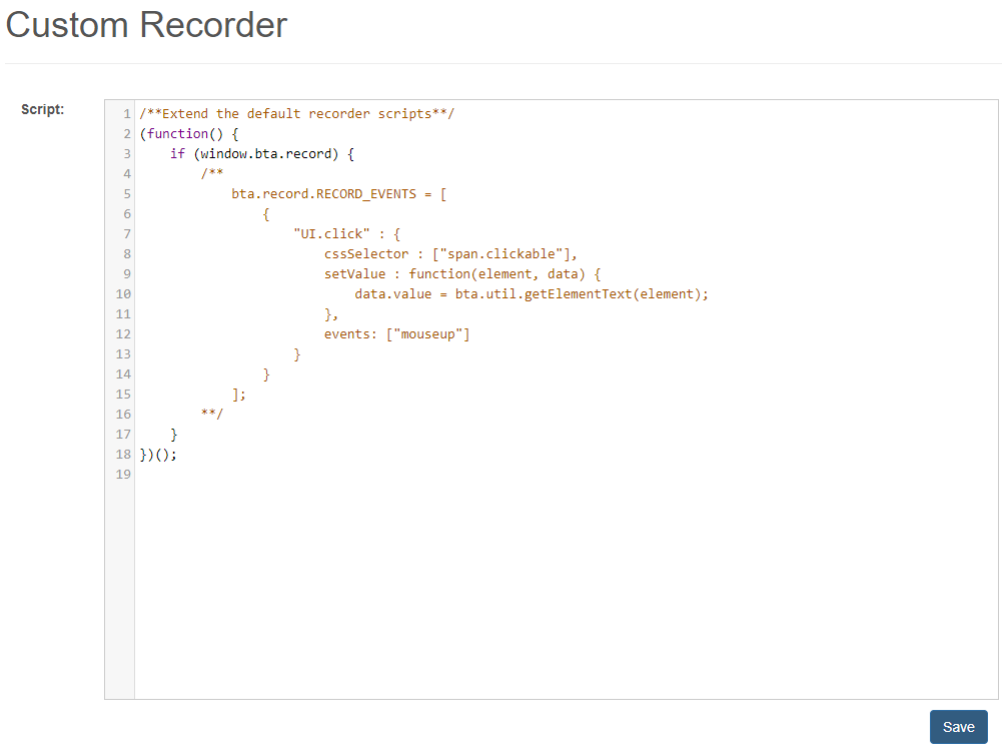
2. The left table of modal lists the selenium girds which do not belong to team, while the right do. And you can select selenium girds in the left table and click  to assign users to the team. Similarly, you can select slelenium grids in the right table and click  to remove users from the team.



Custom recorder

Overview

To support the various modern UI frameworks, we can extend the IDA recorder script. The page can be accessed by the menu **Administration -> Custom Recorder**.Usually we use this feature to provide a hot fix to customer if record feature need to be enchanced.



Uncomment the definition of the **bta.record.RECORD_EVENTS** to enable the script.

|**bta.record.RECORD_EVENTS**|Array of **recorder mapping**|This is the array to maintain the mapping between test command and the target UI control. |**The recorder mapping**|JSON object|The key represents a test command which compose of the command category and command name. For example: **UI.click** stands for the command **click** under category **UI**. The value is **the target UI control configuration**. |**The target UI control configuration**|JSON object|The JSON contains three keys **cssSelector**, **setValue** and **events**. |**cssSelector**|Array of string|The css selector can identify the UI controls. |**setValue**|function(element, data)|**element** is the target UI control DOM element. **data** is the recorded data. |**events**|Array of string|The recorder will fire the corresponding event on the target UI control, and also trigger the **setValue** function.

Start and Stop IDA environment

Starting your environment

Start MySQL

Start the MySQL server by calling the following command:

- Linux: [MYSQL_HOME]/bin/mysqld
- Windows: [MYSQL_HOME].exe

Start Selenium Hub

Start the Selenium Hub by calling the following command:

- Linux: [JAVA_HOME]/bin/java -jar [SELENIUM_HOME]/selenium-server-standalone-3.14.0.jar -role hub -newSessionWaitTimeout 10 -timeout 30 -browserTimeout 30
- Windows: [JAVA_HOME].exe -jar [SELENIUM_HOME]-server-standalone-3.14.0.jar -role hub -newSessionWaitTimeout 10 -timeout 30 -browserTimeout 30

Start Selenium Node

Start the Selenium Node by calling the following command:

- Linux: [JAVA_HOME]/bin/java -Dwebdriver.gecko.driver="[SELENIUM_HOME]/geckodriver.exe" -Dwebdriver.chrome.driver="[SELENIUM_HOME]/chromedriver.exe" -Dwebdriver.ie.driver="[SELENIUM_HOME]/IEDriverServer.exe" -jar [SELENIUM_HOME]/selenium-server-standalone-3.14.0.jar -role node -hub http://localhost:4444/grid/register/ -timeout 30 -browserTimeout 30
- Windows: [JAVA_HOME].exe -Dwebdriver.gecko.driver="[SELENIUM_HOME]/geckodriver.exe" -Dwebdriver.chrome.driver="[SELENIUM_HOME]/chromedriver.exe" -Dwebdriver.ie.driver="[SELENIUM_HOME]/IEDriverServer.exe" -jar [SELENIUM_HOME]-server-standalone-3.14.0.jar -role node -hub http://localhost:4444/grid/register/ -timeout 30 -browserTimeout 30

Start WebSphere Liberty

Start the WebSphere Liberty server by calling the following command:

- Linux: [IDA_HOME]/bin/server start [SERVER_NAME]
- Windows: [IDA_HOME].bat start [SERVER_NAME]

Stopping your environment

Stop WebSphere Liberty

Stop the WebSphere Liberty server by calling the following command:

- Linux: [IDA_HOME]/bin/server stop [SERVER_NAME]
- Windows: [IDA_HOME].bat stop [SERVER_NAME]

Stop Selenium Node

Stop the Selenium Node by terminating the corresponding java process.

Stop Selenium Hub

Stop the Selenium Grid by terminating the corresponding java process

Stop MySQL

Stop the MySQL server by calling the following command:

- Linux: [MYSQL_HOME]/bin/mysqladmin -uroot shutdown
- Windows: [MYSQL_HOME].exe -uroot shutdown

Logging configuration

About this task

In this task, you can runtime set log level in IDA, after that you can see the logs in real time based on the log level changes.

Set log level in IDA

1. Click Administrator tab, then switch to custom log level tab.
2. Change the log level and click save button to save the log configuration.In default it have four levels (DEBUG,WARN,INFO,ERROR).

3. After that you can find the log level is changed in the console,log file etc.

Notes: If you restart the server,the log configuration will restore to orignal configuration.

Monitoring Configuration

About this task

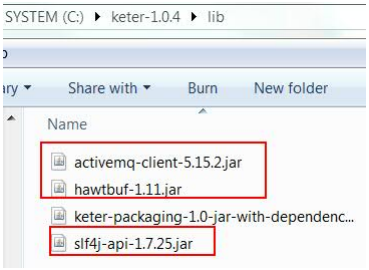
In this task, you will integration BPM DEF with ActiveMQ embedded in IDA, so that IDA monitor can receive BPM performance data and present as chart to IDA users.

Prerequisite

To integrate BPM DEF with with ActiveMQ embedded in IDA, the version of IBM BPM should be **8.6 CF2017.12** or above.

Integrate BPM with IDA

1. Copy below three jar files from IDA release **lib** folder to **%BPM_HOME%/lib/ext** folder.



2. Access IBM BPM Websphere Application Server Administrative Console in web browse and navigate to **Resources -> JMS -> JMS providers**.
3. Click **New** button and enter *ActiveMQ* as **Name**, *org.apache.activemq.jndi.ActiveMQInitialContextFactory* as **External initial context factory**, and *tcp://mqhost:61616* as **External provider URL**.

Name	Value
Name	ActiveMQ
External initial context factory	org.apache.activemq.jndi.ActiveMQInitialContextFactory
External provider URL	tcp://mqhost:61616

Please replace mqhost with the actual IP address where IDA is deployed.

4. Click **Custom properties** in **Additional Properties** panel then click **New** button to add a new property *xa=true* as String type.
5. Click OK button to complete the JMS provider creation.
6. Navigate to Resources -> JMS -> Queue Connection Factories and create a new one.

Name	Value
Provider	ActiveMQ
Name	myQCF
JNDI Name	jms/myQueueConnectionFactory
External JNDI Name	QueueConnectionFactory

7. Navigate to Resources -> JMS -> Queues and create a new one.

Name	Value
Provider	ActiveMQ
Name	myQueue
JNDI Name	queue/myQueue
External JNDI Name	dynamicQueues/myQueue

8. In BPM server, from the *install_root/profiles/Dmgr01/bin* directory, run the following command.

```
wsadmin -lang jython -f <Install_Root>/BPM/Lombardi/tools/def/SampleConfigureJSONEventsToJMS.py
```

9. In BPM server, from the *install_root/profiles/Dmgr01/bin* directory, run the following command.

```
wsadmin -lang jython -f <Install_Root>/BPM/Lombardi/tools/def/SampleReloadDEF.py
```

10. After you run the sample script, a **defconfig.xml** file is created in the *dmgr_profile_home/config/cells/cellName* directory. Edit this file to update the value of *JMS_QUEUE_JNDI* to **queue/myQueue**, and the value of *JMS_QUEUE_CF_JNDI* to **jms/myQueueConnectionFactory**. Please note that the **jms/myQueueConnectionFactory** and **queue/myQueue** are the JDNI name created in step #6 and #7.
11. In BPM WAS Administrative console, from System administration -> Nodes, select all nodes then click **Full Resynchronize** button to synchronize the changes to node profiles.
12. Restart BPM.

Architecture

Reference Architecture

Integration Architecture Reference

- IBM Operational Decision Manager will be used to replace OOTB static analysis POJO engine so that all static analysis rules could be customized and managed outside IDA solution.

Test Commands

This document refers to the Command Editor options and not JavaScript API to call a command. JavaScript API for the test case step can be viewed by clicking the label for the step. API inputs may not be named exactly the same as Command Editor fields. For example, **Assignee** in the Dialog box corresponds to a **user** in the API.

There are five(5) categories of commands:

- BPM
- BPM Assertion
- UI
- UI assertion
- Utility

BPM

This category of commands are for calling server side components. There are a lot of common input fields. Below are some statements which are common for most of the commands in this category.

Assignee: When specified, the command is executed by the specified BPM user. Drop down lists all *BPM user names* defined in the BPM Configuration the Test Project is connected to. More details refer to **BPM Configuration/Add user to a BPM server**.

Exception: is an exception case. Checked means we are expecting to get an error for the test case. Unchecked means we are expecting to get a normal result from the test case.

Comment: Optional. Not part of any command and can be used for documenting test case step.

Params: Json format. If the template of parameters is rendered successfully by selecting service, the field is required. The parameters are defaulted based on the service signature. The template of parameters includes keys of data and you only need to fill in value. *e.g:*

```
{
"variable_name": "value",
"variable2_name": "value2"
}
```

The icon will appears below *Expected Output* input box when the wrong *Params* is filled out after click button in the Command Editor Dialog. The error message can be got by hovering on the icon.

Expected Output: Used for validating command response. In some cases, there is a refresh button, which can be used to get default expected results. Make sure to fill out params before clicking refresh. If refresh does not work, execute the case step with out expectedOutput set. If passed, edit the step and click on button in the Command Editor Dialog to capture the results.

TaskName: Drop down lists all task names in the Process Application of the Test Project which is connected to.

Json Path: Json path to a variable in the output json data.

Variable Name: Variable name to save the value the json path points to. This variable can be accessed via ${variable_name} in later commands in this test case.

Command Name	Parameters
loginPortal	Assignee: select a user defined in BPM Configuration.
loginBPMoC	Assignee: select a user defined in BPM Configuration.
openInstanceDetail	Assignee: select a user defined in BPM Configuration.
startProcess	Process : name of the process to be started. Params: : (optional) input parameters in JSON format. Tip: when creating a case from the Test case menu for a Process, the default parameters are generated.
startHumanService	Service : name of a human service to be started.Select from drop down. Params: human service input list. Only simple input arguments are supported.
startExposedHertiageHumanService	Service : name of a human service to be started.Select from drop down. Params: human service input list. Only simple input arguments are supported.
startAjaxService	Exception: is an exception case. Service: service name, select from drop down. Params: service input list. Expected Output: expected service output in JSON.Tip: first, execute the case step with out expectedOutput set. If passed, edit the step and click on button in the Command Editor Dialog to capture the results.

Command Name	Parameters
startSystemService	Exception: is an exception case. Service: service name, select from drop down. Params: service input list. Expected Output: expected service output in JSON.Tip: click on button in the Command Editor Dialog to capture the example output.
startSQLQuery	SQL: sql statement. maxRows: max rows Data Source Name: data source JNDI name as defined in target BPM environment. Default dataSourceName is jdbc/TeamWorksDB. Expected Output: expected output. Json Path: json path to a variable in the output.
startUCA	UCA: uca name, select from drop down. Params: UCA inputs.
startServiceFlow	Exception: is an exception case. Service: select service name from drop down. Params: service input list. Expected Output: expected service output in JSON, click on refresh to auto generate.
startAdHoc	Ad Hoc Name: Ad Hoc name, select from drop down.
runAdHocActivity	Ad Hoc Activity Name: ad Hoc activity name,select from drop down.
runTaskByDisplayName	Task Name: task name, select from drop down.
runTaskByActivityName	Task Name: task name, select from drop down.
assignTask	Task Name: task name, select from drop down. To User: select from drop down list of users defined in IDA/BPM Admin for target BPM server.
claimTask	Task Name: task name, select from drop down.
finishTask	Task Name: task name, select from drop down. Output: task output, json format. Set this value to the output of the task.
addInstanceId	Instance ID: instance id of the process.
getTaskIdByName	Task Name: task name, select from drop down.
getLatestInstanceIdByProcess	Process Name: Process name to get latest instance. Select from drop down.
getInstanceIdByInstanceName	Process Name: Process name to get instances. Select from drop down. Instance Name: instance name of the instance id.
getInstanceIdByTaskName	Process Name: Process name to get tasks. Select from drop down. Task Name: task name.
getInstanceIdByTaskUrl	Assignee: select a user defined in BPM Configuration.
getInstanceIdByBusinessData	processName: process name, select from drop down. Business Data Alias: business data alias name. Business Data Value: business data value.
fireTimer	Token: timer name selected from drop down.
startRestApi	API Name:rest api name. Parameter: rest api parameter template. Expected Value: the expected rest api response.

BPM Assertion

This category is for asserting server side components. Here are statements of common input fields.

Assignee: When specified, the command is executed by the specified BPM user. Drop down lists all *BPM user names* defined in the BPM Configuration of the Test Project which is connected to. More details refer to **BPM Configuration/Add user to a BPM server**.

Task Name: Name of the task that needs to be asserted. Drop down lists all *Task names* of current process.

Comment: Optional. Not part of any command and can be used for documenting test case step.

| Command Name | Parameters |Description | | ——————————— |————————————————————— ——- ——- ——- - ——- ——- |
| assertProcessInstanceStatus |Instance Status: expected status of the current process instance.|Check current process instance status.
Possible values:
Active
Completed
Failed
Terminated
Suspended|
| assertProcessInstanceData |Expected Value: JSON object, expected value of the data.
Notes:
Before adding this step, run the test case and look at the report. In the **Process Instance** section, click on the **click to show full data** link.

Copy the shown JSON into the **Expected Value** as data template. Then change the value as you need. If your data has auto generated fields, like current dates, instance ids or some other unique identifiers, they have to be removed from the JSON.| Assert the data of current process instance. Can only be used in a context of process instance execution.|
| assertTaskAssignmentByUser | Task Name: name of the task that needs to be asserted. Select from drop down.| Check user is assigned (claimed) to the task. Can only be used in a context of process instance execution. | assertTaskStatus | Task Name: name of the task that needs to be asserted. Select from drop down.
Task Status : expected status of the task.| Check the status of a BPM task for current process instance.
Valid task status values:
Received
Closed|
| assertNextTaskName |Task Name: name of the task that needs to be asserted. Select from drop down.| Check the task is received.| | assertTaskNotGenerated |Task Name: name of the task that needs to be asserted. Select from drop down.| Check the task is not generated. |
| assertTaskData |Task Name: name of the task that needs to be asserted. Select from drop down.
Expected Value: expected value of the data.
Notes:
Before adding this step, run the test case and look at the report. In the **Task Data** section, click on the **click to show full data** link.

Copy JSON value into the **Expected Value** as data template. Then change the value as you need. If your data has auto generated fields, like current dates, instance ids or some other unique identifiers, they have to be removed from the JSON.| Check the task data of current process instance.It checks the variables saved in BPM Task execution context. Use UI Assertions to check on page contents during the coach execution.|
| assertServiceData |Expected Value: expected value of the data.
Notes:
Before adding this step, run the test case and look at the report. In the **Service Data** section, click on the **click to show full data** link.

Copy JSON value into the **Expected Value** as data template. Then change the value as you need. If your data has auto generated fields, like current dates, instance ids or some other unique identifiers, they have to be removed from the JSON.| Check the currenct service data. |

UI

In most cases, the UI commands are recorded using the IDA plug in. If you find a need to add a command manually, for example, validation command, look at the recorded command for the element and copy the location information from it.
e.g: Any one or combination of the recorded values can be used to identify the control: Control ID, Element ID or XPATH. If Label is unique on the page, it can be used as well. Enough location arguments are required to uniquely identify an element on the page. **IDA** cycles through available ids until it finds the element. For example, if you recorded a select step and later moved the control around on the page, the Control ID and XPATH are probably different from the recorded step, but Element ID is still the same.

Here are statements of common input fields.

Assignee: When specified, the command is executed by the specified BPM user. Drop down lists all *BPM user names* defined in the BPM Configuration of the Test Project which is connected to. More details refer to **BPM Configuration/Add user to a BPM server**.

Value: The value of the control.

Contorl ID: The ID of the control. Control ID is the PATH control IDs leading to the selected control id.

Label: The label of the control.

Element ID: The ID of the element.

Element CSS: The CSS of the element.

XPATH: The xpath of the control.

Comment: Optional. Not part of any command and can be used for documenting test case step.

Notes: *The element is first obtained through the Control ID. If is failed, the label, Element ID and Element CSS are then used in turn. Finally is XPATH. If is successful, the rest will not be used.*

Command Name	Parameters
bpmFileDropzone	Value: the value of the BPM file. Control ID: the control id of BPM file drop zone. Default value: the value of default.
bpmFileUploader	Value: the value of the BPM file. Control ID: the control id of BPM file that needs to be Uploaded. Default value: the value of default.
checkbox	Refer to common statements.
click	Value: Click text. Type: *Link* or *Button*.
coachControl	value type: select type from drop down. Default Value: default value of the control.
confirmOK	N/A
confirmCancel	N/A
doubleClick	Value: Click text.
file	Value: file name that needs to be uploaded.
open	URL: the URL of browser.
close	Refer to common statements.
radio	Refer to common statements.
saveCoachControl	Key : Key is a name of a variable, which can later be referenced as ${key} in a value field of any command. Property : When the control binding complex object, you can get the value form complex object by property. *eg:* *bindingObject.value*
saveText	Key : Key is a name of a variable, which can later be referenced as ${key} in a value field of any command.
select	Refer to common statements.
selectWindow	Info: window information.
text	Refer to common statements.
waitElement	Type: select the element type from drop down. *Visible*, *Hidden*, *Enabled* and *Disabled*. Timeout: the maximum time that wait for the element.
waitTextPresent	Text: t that waits for displaying on the page. Timeout: the maximum time that wait for the text.

UI Assertion

Record a command to populate the field you want to assert. Then edit the generated test case step and change command to assert. The edit would pick up matching fields from the recorded command.

Here are statements of common input fields.

Assignee: When specified, the command is executed by the specified BPM user. Drop down lists all *BPM user names* defined in the BPM Configuration of the Test Project which is connected to. More details refer to **BPM Configuration/Add user to a BPM server**.

Contorl ID: The ID of the control. Control ID is the PATH control IDs leading to the selected control id.

Label: The label of the control.

Element ID: The ID of the element.

Element CSS: The CSS of the element.

XPATH: The xpath of the control.

Expected Value: The expected value of the control.

Comment: Optional. Not part of any command and can be used for documenting test case step.

Notes: *The element is first obtained through the Control ID. If is failed, the label, Element ID and Element CSS are then used in turn. Finally is XPATH. If is successful, the rest will not be used.*

Command Name	Parameters

Buttons	Texts: texts of buttons that you want to render.
	e.g:
	If you want to check that all the buttons are rendered in this form, you can fill like this:
	`Save`
	`Cancel`
	Notes: texts are separated by the line break.

assertCheckbox	Expected Visibility: the visibility of the control. Select from drop down. *EDITABLE*, *READONLY,* and *HIDDEN.*
	Checked Labels: labels of checkboxs that are expected to be checked.
	e.g:
	To verify the options in this page, fill in like this:
	`Create Position Request CSHS`
	`GM Approval CSHS`
	Expected Options: labels of checkboxs that are expected to be rendered.
	e.g: If you want to check that all the checkbox options are rendered in the figure above, you can fill in like this:
	`Create Position Request CSHS`
	`Find Candidates CSHS`
	`GM Approval CSHS`
	Notes: labels are separated by the line break.

assertCoachControl	Expected Value Type: the value Type of the control, select from drop down. *String*, *Number*, *Date(MM/dd/yyyy)* and *Boolean*. Expected Visibility: the visibility of the control. Select from drop down. *EDITABLE*, *READONLY,* and *HIDDEN.*
assertDatePicker	Expected Date: the expected date of the DatePicker.
assertElement	Expected visibility: the visibility of element. Select from drop down. *EDITABLE*, *READONLY,* and *HIDDEN.*
assertInputText	Expected Max Length: expected max length of the text.
assertOutputText	Refer to common statements.
assertRadio	Checked Label: the radio box is checked or not. Expected Options: labels of radio boxes that are expected to be rendered. Refer to the statements of *Expected Options* in **assertCheckbox**.
assertSelect	Expected Value: the expected value of the control. If is multiSelect, the value is separated by the line break. Expected Options: labels of select options that are expected to be rendered. Refer to the statements of *Expected Options* in **assertCheckbox**.
assertSwitch	Refer to common statements.
assertTableContent	Column Name: the name of the column where the value needs to be asserted. Row Number: the number of row where the value needs to be validated. Expected value: the cell value that needs to be checked.
assertTableRows	Expected Rows: the expected total number of rows in the table.
assertTextarea	Refer to common statements.
assertTextNotPresent	Text: the text that is not expected to appear on the UI.
assertTextPresent	Text: the text that is expected to appear on the UI.
assertValidationPassed	Refer to common statements.

Utility

This category provides utility commands. Here are statements of common input fields.

Assignee: When specified, the command is executed by the specified BPM user. Drop down lists all *BPM user names* defined in the BPM Configuration of the Test Project which is connected to. More details refer to **BPM Configuration/Add user to a BPM server**.

Comment: Optional. Not part of any command and can be used for documenting test case step.

Command Name	Parameters
addContext	Key: name of key. Value: name of value.
dateString	Key: key for save data string. Days: the number of days from the current date that you want to convert to a string. Negative number of days means before today, while positive number of days means after today. *e.g:* **-1** means generating string of the date of yesterday. Format: Format of date.
debug	Log: log text will be shown in console.
putContext	Key: name of key. Value : the name of value.
wait	Time: the seconds to be wait.
randomString	Length : length of the generated random string.
randomNumber	Length : length of the generated random string.

Javascript API

OOTB Javascript API

JS API	Parameters	Comments
bta.util.callback		Default callback for Javascript command
bta.util.debug	message	add debug info on the browser console
bta.util.putContext	key, value	add key,value into the memory
bta.util.getCoachView	JSON	Get the coach view object
bta.util.getElementText	DOM Element	Get the DOM element text
bta.util.assertTrue	condition(boolean), message(string)	Asserts that a condition is true. If it isn't it throws an error with the given message.

JS API	Parameters	Comments
bta.util.assertFalse	condition(booelan), message(string)	Asserts that a condition is false. If it isn't it throws an error with the given message.
bta.util.assertEquals	expected(string), actual(string), message(string)	Asserts that two objects are equal. If they are not, an error is thrown with the given message.
bta.util.throwError	message(string), retry(boolean, default is true)	Throw error with the given message

Notes
Beside these OOTB JS API, user can also use standard BPM coach view JS API when writing the custom JS. e.g - var rows = tableCoachView.context.element.querySelectorAll("div.SPARKTable table.table tbody tr");

IDA System variables

The following table lists all system variable names and JS comand could not use these as variable name.

|————-|:————-:|:——| |instanceId|bpmUser|processConfig|bpmConfig |engineConfig|parameters|taskName|toUser |output|token|processName|businessDataAlias |businessDataValue|instanceName|adHocActivityName|adHocName |service|params|toolkitName|exception |expectedOutput|variableName|jsonPath|apiName |errorMessage|expectedValue|sql|maxRows |dataSourceName|expectedRows|uca|key |value|days|format|time |length|templatePath|script|caseId |serverHost|next|process|

REST APIs

IDA REST API reference.

Pipeline

- **`POST /rest/v1/pipeline/build?pipelineId=:id` Trigger build by ID**
- **`POST /rest/v1/pipeline/build?pipelineName=:name` Trigger build by name**
- **`GET /rest/v1/builds/:buildId` Get build status**

Trigger build by id

URL:

`/rest/v1/pipeline/build?pipelineId=:id`

Method: `POST`

Parameters:

Name	Required	Description
`pipelineId`	yes	Pipeline ID. You could get the pipeline ID from the URL path of pipeline dashboard page.
`userToken`	yes	User token. You could get the token from user profile page.
`snapshotAcronym`	no	BPM snapshot acronym name. If it's set, pipeline build will bypass the Snapshot step, and the Test/Checkstyle/Deployment steps will use this snapshot.
`twxFileName`	no	The BPM installation package file name of Process APP.

Example:

Request

```
curl -X POST http://<SERVER>/ida/rest/v1/pipeline/build?pipelineId=1 -k --data-urlencode "snapshotAcronym=0.00.96" --data-urlencode "userToken=kBYf/PfN6B5aQsXyEZaNHBH/aIezRtwjm4w5EcEEqy+CfBmE/7OCQ6HGYnaSKBFw"
```

Response

```
{
    "buildId": 1314,
    "pipelineId": 1,
    "status": "RUNNING"
}
```

Trigger pipeline build by name

URL:

`/rest/v1/pipeline/build?pipelineName=:name`

Method: `POST`

Parameters:

Name	Required	Description
`pipelineName`	yes	Pipeline name.
`userToken`	yes	User token. You could get the token from user profile page.
`snapshotAcronym`	no	BPM snapshot acronym name. If it's set, pipeline build will bypass the Snapshot step, and the Test/Checkstyle/Deployment steps will use this snapshot.

Name	Required	Description
`twxFileName`	no	The BPM installation package file name of Process APP.

Example:

Request

```
curl -X POST http://<SERVER>/ida/rest/v1/pipeline/build?pipelineName=pipeline%20sample -k --data-urlencode "snapshotAcronym=0.00.96" --
data-urlencode "userToken=kBYf/PfN6B5aQsXyEZaNHBH/aIezRtwjm4w5EcEEqy+CfBmE/7OCQ6HGYnaSKBFw"
```

Response

```
{
    "pipelineName": "pipeline sample",
    "buildId": 1315,
    "status": "RUNNING"
}
```

Get build status

URL:

```
/rest/v1/builds/:buildId
```

Method: **`GET`**

Parameter:

Name	Required	Description
`buildId`	yes	Build ID. You could get the build Id from the reponse of pipeline build API.

Example:

Request

```
curl http://<SERVER>/ida/rest/v1/builds/1314
```

Response

```
{
    "report": "http://<SERVER>/ida/rest/v1/pipelines/1/builds/1314",
    "buildId": 1314,
    "status": "RUNNING"
}
```

Response Status

Each response will be returned with one of the following status:

Build Status:

- `RUNNING` The pipeline build is running
- `FAILED` There was a problem with pipeline build (step failed, etc.)
- `SUCCESS` The pipeline build was successful

Other Status:

- `unauthorized` The token provided do not have permission to access the requested resource
- `pipeline not found` An attempt was made to access a pipeline that does not exist
- `build not found` An attempt was made to access a build that does not exist

Migrating and updating IDA Application

Preparing your migration

To prepare your migration, take the following steps:

1. Download the lastested IDA version.
2. Stop the libery server.
3. Stop the mysql server.
4. Backup the mysql db.

Step 1: Update DB

To update DB, take the following steps:

1. Start the mysql Server.
2. You can find migrate-mysql*.sql in the sqlfolder. Copy the corresponding version upgrade sql into clipboard.
   For example: If your previous version is v2.2.2.You need to pick below sql script to run one by one.Make sure the migrate sql version name is bigger than your previous version.

- migrate-mysql-v2.2.2-v2.2.3.sql
- migrate-mysql-v2.2.3-v2.2.4.sql
- migrate-mysql-v2.3.0-v2.3.1.sql
- migrate-mysql-v2.3.1-v2.3.2.sql

3. Connect to the MySQL server and use IDA database. Execute the script for each consecutive version.

```
mysql> use IDA ;
```

```
mysql> paste your sql here
```

Step 2: Update IDA.war

For IDA version migration,you need to update IDA.war, take the following steps:

1. Find ida.properties under conf folder in the previous version.
2. Reconfigure ida.properties file in the new version.Make sure you have copy the every propertie value from old version to new version. After you finish the properties value changes,you need to double check these properties values are set in new version files.
   3.Click the package.bat command to repackage the ida-web.war file.It will update application-prod.yml
3. Remove all the files from wlp installation locationfolder.
4. Copy the ida-web.war (which generated in step3) into the wlp installationfolder.
5. Start the Liberty Server.

Notes
Please not overwrite ida.properties from previous version, since we might add new some new property name in some versions.
You can check the application-prod.yml in ida-web.war to make sure these setting are applied.

Samples

Samples

The samples are available on Git samples repository. - Testing samples list - Testing sample (BPM, assertion and utility commands)
- UI command sample for BPM v8 Coach UI - Coach Sample - UI command sample for BPM responsive Coach toolkit - Responsive Coach Sample - UI command sample for BPM UI toolkit - BPM UI Sample - UI command sample for BP3 toolkit - Write a JS custom command - JS custom command sample - Write a Java custom command - Java Command Sample - Headless sample - How to test the headless process application - BPMoC sample - How to test the BPMoC process

Online demos

1. YouTube demos: IBM Deployment Accelerator for Business Automation Workflow - IDA Playlist

Known Limitation

- The IDA plugin only supports Firefox and Chrome.
- The JS window.alert function can't be supported by IDA plugin, so we disable it in IDA plugin.
- IDA firefox plugin doesn't support self-sign certification very well, please use http protocol if you want to use IDA firefox plugin.
- BPM Deployment feature only supports Linux server via SSH (or install SSH server on Windows server).
- Complex parameters can not be passed to the human service.
- Global variables declared in Server Managed files .JS are flagged as "was used before" it was defined in checkstyle report.
- The file upload commands UI.file and UI.bpmFileUploader can't be replayed on IDA Plugin due to browser security issue, but these commands can work on Selenium Grid. So you can record the file upload commands by IDA Plugin, but replay them on Selenium Grid.
- The command UI.close is disabled in IDA plugin which may caused issue in plugin.
- BP3 Richtext control replay not working if no value is binding for this control.

Best Practices

Long load time AJAX call

- As we know, the modern web application always load data by AJAX call. Sometime, it may needs a long time to get the AJAX call result. The automation test case become not stable in this case. It's because that, the test case run the test steps without waiting the AJAX call end.
- Normally, there is a loading indicator on the HTML to indicate a AJAX call is completed or not. We could resolve the problem by adding the **waitElement** command after the AJAX call test step. So that the system will wait a certain seconds until the loading indicator element becomes **visible/hidden/enabled/disable**.
- The **waitElement** command:
  - **Type**: The value could be Visible, Hidden, Enabled and Disable.
  - **Timeout**: The waiting seconds.
  - **Element ID/Element CSS/XPATH**: These values can be used to locate the AJAX loading indicator HTML element on the UI page.

IDA plug-in troubleshooting

- When you run IDA plug-in for replay you might meet issue for the plug-in. You should see the step with highlighted red color.When you move the mouse the red icon,it will show the detail issue.

You can use below ways to collect logs of plug-in.

1. You can use the browser console to see the logs for IDA recorder plug-in console.
2. You can use the browser console to see the logs for UI console.
3. Manage the extension of IDA plug-in.Switch the "Developer mode" on the right top page.Then click the Inspect views "background page" to see background.js console message.
4. You can also check server side console to see any logs with this step.

Frequently asked questions

FAQ about IDA

1. FAQ: Where can I get information about hardware and software requirements of IDA?
   - Answer: See System requirement doc
2. FAQ: Could IDA be used to do performnance testing?
   - Answer: No at the moment.
3. FAQ: Can Checkstyle or Testing component be integrated individually with the existing external CI/CD pipeline?
   - Answer: YES, you could define Testing only, Checkstyle only pipeline and then leverage REST APIs for integration. More detail please refer to Architecture doc and API doc
4. FAQ: Could IDA use Oracle or SQLServer as DB beside DB2 and Mysql in supported list?
   - Answer: IDA only tests on DB2 and Mysql by now.It need some consultancy hours to support other RDBMS.
5. FAQ: Could IDA WAR be deployed on the BPM WAS server?
   - Answer: NO. The BPM WAS server COULD be only used by BPM application if it's not a license WAS server.
6. FAQ: Could we use IE to record testing?
   - Answer: NO. the recording feature only supports on Chrome and Firefox. IE could be used to replay the recorded test case. Just need setup a Selenium server on Windows to run regression testing.
7. FAQ: Could we import Selenium test projects to IDA?
   - Answer: NO.
8. FAQ: Does IDA support BPM adv (BPEL) testing and static analysis?
   - Answer: NO. The testing and checkstyle of IDA only supports BPM std(BPMN).
9. FAQ: Does IDA support IBM BAW v18?
   - Answer: YES. Refer to *Software Prerequisites* in System requirement doc
10. FAQ: Can we customize the rules of Checktyle?
    - Answer: YES. Refer to doc Customize Checkstyle rules and Architecture doc about integration.
11. FAQ: Does the CheckStyle feature detect issues that cannot be detected by the Process Designer code validation
    - Answer: YES. Checkstyle is BPM code static analytic.The PD validation is to do "compiling validation".
12. FAQ: Could IDA test process definitions in toolkits or just the ones in process applications?
    - Answer: YES. The user could select toolkit when creating test case. See doc
13. FAQ: If the team built some AngularJS code inside the coaches, can IDA support its testing?
    - Answer: YES. We could use Recorder extention to capture all UI controls if the OOTB recorder does not record it correctly.
14. FAQ: What is the customer gets under the contract: source code of IDA or the packaged compiled code??
    - Answer: The packaged compiled code w/o source code.
15. FAQ: Does IDA support heritage coaches which is wildly used in BPM v8.x?
    - Answer: YES.
16. FAQ: Does IDA support BPMoC?
    - Answer: YES.
17. FAQ: Can a sample report be shared with us so we want to look at it more closely off-line to understand what checkstyle can analysis?
    - Answer: YES. You could share one TWX file with us and we coudl use IDA to generate a offline report.
18. FAQ: Is there technical documentation available for IDA?
    - Answer: All have been published on Git doc
19. FAQ: Is there number of users limitation on the IDA?
    - Answer: No limitation on # of users. The IDA asset license is granted to setup one IDA env.
20. FAQ: How does Checkstyle of IDA compare with IBM App Scan ?
    - Answer: The app scan only check for security vulnerability and the checkstyle is do statci code analysis in BPM TWX.
21. FAQ: Shall I record all test cases again if I add a new field or change its position?
    - Answer: You could use group command to manage test cases. In that case, you could just manualy modify the group command if adding one new field. Or record group command steps again and not need update all test cases one by one.
22. FAQ: Does IDA support migration from versions which are not previous to the current release? For example, 2.2.1 to 2.2.4?
    - Answer: YES. though the intermediary sql scripts must be executed for each consecutive version we skip. Refer to step one in migration doc
23. FAQ: Why can't I launch the BP3 human services on firefox grid?
    - Answer: The firefox selenium driver that we used(geckodriver-v0.23.0) cannot launch the BP3 human services when the selenium version is 3.14.0. This will be fixed in the next release.
24. FAQ: Do we need the BPM admin to execute WAS commands permanently or we can use another user with similar/less permissions in configuration?
    - Answer: We need WAS admin role to execute WAS commands to do deployment if the version is BAW v18 less. If it's BAW v18.0.0.2 or better, there is no need to configure WAS Admin in IDA.
25. FAQ: There are only user and admin two roles in doc. Can we define more roles?
    - Answer: Yes. e.g You need an infrustrure ADMIN role to do all servers configuartion. It could be done thru custimzation for role permission control.
26. FAQ: Can we use a different user for the REST API configuration?
    - **Answer: YES. Please make sure the user is in group either tw_authors or tw_admins.**

title: "Demos" category: references date: 2019-01-17 15:17:55 last_modified_at: 2019-07-29 16:54:00 order: 10 —

Demos

YouTube demos

1. IBM Deployment Accelerator for Business Automation Workflow - IDA Playlist
2. IDA Introduction
3. IDA Tutorial: BPM Testing - Creating and Generating Process Test Cases
4. IDA Tutorial: BPM Testing - Recording and Replaying Process Test Cases
5. IDA Tutorial: BPM Testing - Creating and Replaying Service Test Cases
6. IDA Tutorial: BPM Testing - Refactoring Your Test Cases with Group Commands
7. IDA Tutorial: Static Analysis and Code Review for BPM
8. IDA Tutorial: Pipeline - Building Your CI/CD Pipeline for IBM BPM in 5 Minutes
9. IDA Tutorial: Pipeline - REST API Introduction
10. IDA Tutorial: Pipeline - Integrating IDA with Jenkins
11. IDA Tutorial: Configuration - Introduction of Permission Management
12. IDA Tutorial: Configuration - Introduction of Team Management