method Call analyzer tool - USER GUIDE

Contents

[installation 2](#_Toc474505373)

[Configuration 3](#_Toc474505374)

[Run the tool 4](#_Toc474505375)

[Software requirements 4](#_Toc474505376)

[Running the class finder tool 5](#_Toc474505377)

[Running the single-level call analyzer tool 6](#_Toc474505378)

[Running the multiple-level call analyzer tool 6](#_Toc474505379)

# installation

This tool works over the generated class files. Usually the class files are located in the **bin** folder (see Figure 1) or the under the **src** folder (see Figure 2), followed by the folder structure of the packages in the module.

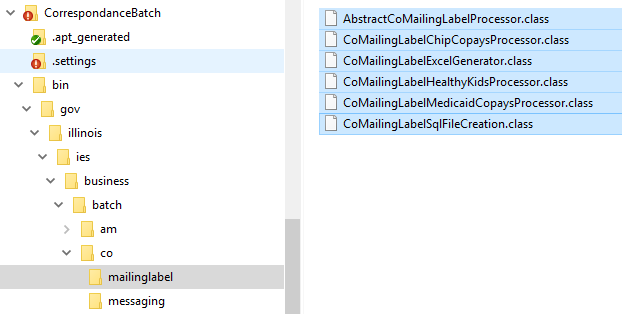


Figure 1. Class files under /bin/ folder

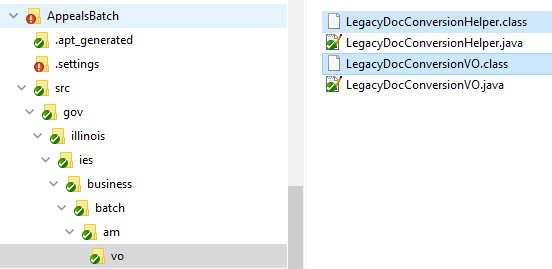


Figure 2. Class files under /src/ folder along with source files

1. Download the zipped folder containing the tool files (method-call-analyzer-workdir.zip) into a known location (the **<working directory>** folder)
2. Navigate to the new created folder. This should have the folder structure shown in Figure 3.
3. The tool uses some libraries to work properly. The list of libraries is shown in Figure 4.

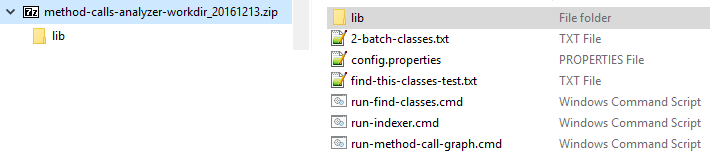


Figure 3. Folder structure of the tool

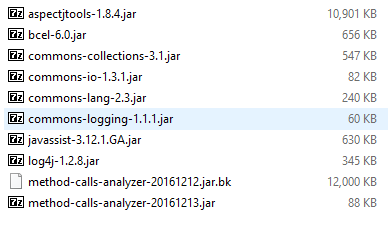


Figure 4. List of libraries used in the analyzer tool

## Configuration

To be able to run the tool you have to set some configurations first.

1. Edit the configuration file you want to use **<config\_file>**. For example **config.properties**
   1. **Required.** Change the name of the file that contains the entry classes (this will be the **<file-with-entries>**). **<file-with-entries>** must exist inside the **<working directory>** folder or an absolute path must be provided. See the example in the current configuration file.

entryFile.path=2-batch-classes.txt

* 1. Change the project name to IES or ABE.

# Options: IES (default), ABE

project.name=ABE

* 1. Add the classes and/or methods you want to analyze. Use the format in the next listing.

full\_qualified\_class\_name|method\_signature(optional)|jobId(optional)

# Example of a class as entry

full.package.name.UnitScheduleBO

#...SHUnitScheduleControllerEJBBean ... This is a commented line

# Examples of methods as entry. Check the format of the signature.

full.package.name.DsBusinessRulesBO**|valuateDSNAP(java.util.Map)**

# No signature will look for all overloaded evaluateDSNAP methods

full.package.name.DsBusinessRulesBO**|evaluateDSNAP**

full.package.name.DsBusinessRulesBO**|evaluateDSNAP|jobId**

* 1. **Required.** Update the location of the root folders with your own. One for each module.

class.root.folders= \

C:/ProjectILIES/ILIES\_BATCH/CCD \

C:/ProjectILIES/ILIES\_BATCH/FrameworkEJB \

C:/ProjectILIES/ILIES\_BATCH/SharedApp \

C:/ProjectILIES/ILIES\_BATCH/AppealsBatch \

C:/ProjectILIES/ILIES\_BATCH/AppealsEJB \

C:/ProjectILIES/ILIES\_BATCH/BenefitMgtBatch \

C:/ProjectILIES/ILIES\_BATCH/BenefitMgtBO \

C:/ProjectILIES/ILIES\_BATCH/BenefitMgtEJB \

C:/ProjectILIES/ILIES\_BATCH/CorrespondanceBatch \

C:/ProjectILIES/ILIES\_BATCH/CorrespondanceBO \

# Add as many folders as needed

* 1. **Optional.** Add, remove or update from the list of excluded packages to be analyzed. These are those packages which are not going to be included (nor analyzed) as part of the report.

packages.excluded= \

gov.illinois.ies.business.entities \

gov.illinois.fw.batch.entities \

gov.illinois.framework.exceptions \

gov.illinois.fw.business.exceptions

gov.illinois.fw \

gov.illinois.framework

# Add as many packages as needed

* 1. **Optional.** Look for comments inside the configuration file about more advanced settings.

# Run the tool

This section explains the *how to* for running the tool including the software requirements to do so.

Must have:

* **<working directory>** as the folder where the tool has been unzipped.
* **<file-with-entries>.txt** as the file containing the entry points.

The tool includes three commands that help to perform the analysis.

* **Class finder**: run-find-classes.cmd
* **Single-level method analyzer**: run-indexer.cmd:
* **Multiple-level method analyzer**: run-method-call-graph.cmd

## Software requirements

You must first have to ensure some software requirements in order to be able to run the tool.

1. Java 6+. To meet the JVM version dependency for all the libraries.
2. Set the *JAVA\_HOME* and/or append the Java *bin* directory to the Path system environment variable. See Figure 5.

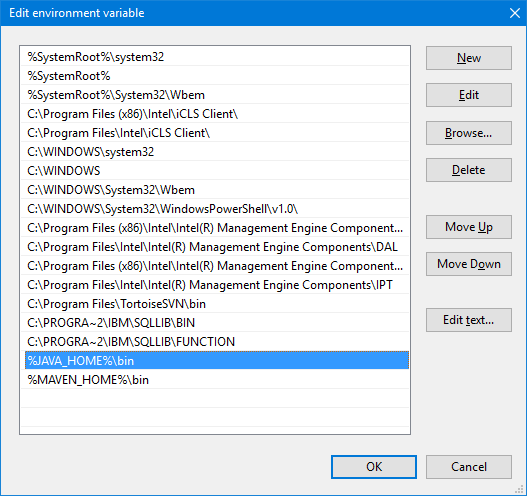


Figure 5. Configuration of the Path environment variable

1. Test the new environment variable by running >java -version. See figure 6.

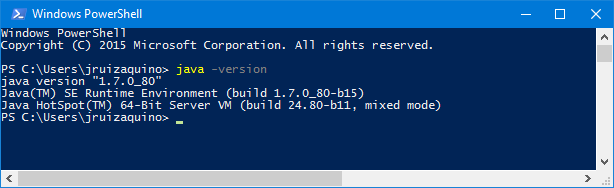


Figure 6. Displaying the installed Java version in command line

1. MS Excel or any text editor to open the report files with extension .csv.

## Running the class finder tool

This tool will help to find the full qualified class name for a list of simple class names or Java base names.

Given a list of no-qualified class names, the tool finds the full qualified class names inside the project’s classes.

1. From command line run **> run-find-classes.cmd <file>.txt <config\_file>**
   * **<file>.txt** Pass the location of the file containing the class names you want to search for full qualified class names.
   * Optional. <config\_file> Default value is config.properties
2. It will generate a result with the name **<file>-result\_classes.txt.** Look at the end of the resulting file for warnings and statistics.

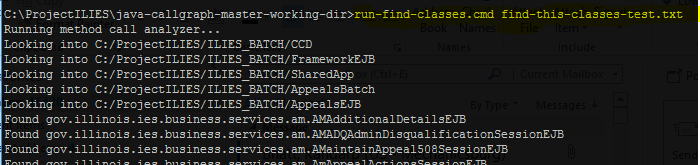


Figure . Running the class finder tool

As an example, take a look at the files *find-this-classes-test.txt* and *find-this-classes-test-result\_classes.txt*.

## Running the single-level call analyzer tool

This tool will analyze the immediate-level calls from a method.

1. From command line run **> run-indexer.cmd <config\_file>**
   * Optional. <config\_file> Default value is config.properties
2. It will generate multiple files with the name *methodRegistry-<FILE\_NUMBER>.csv.* See Figure 8.

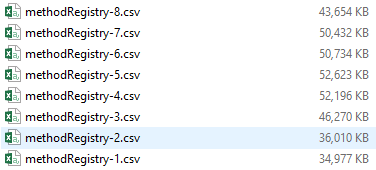


Figure . Results of single-level analysis

## Running the multiple-level call analyzer tool

This tool will analyze multiple-level calls from a method. This means, it will analyze recursively the calls from a method.

1. From command line, run **<working directory> run-method-call-graph.cmd <config\_file>**
   * Optional. <config\_file> Default value is config.properties
2. The tool will generate a folder **<working directory>/report/<file-with-entries>** with all the reports. The report will have the next formats (see Figure 9):
   * One per analyzed entry with a name like *package.name.ClassName.methodName-visited\_methods.csv*
   * Multiple files with the summarized results with a name like *All\_Classes-visited\_methods-<FILE\_NUMBER>.csv*
   * A statistic report with the name *global-statistics.txt*

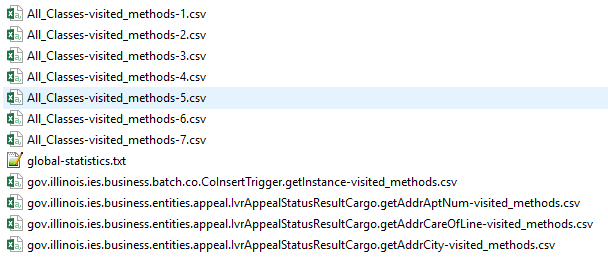


Figure . List of report files

1. It also will generate a *log.txt* under your working directory.