Kiuwan Local Analyzer CLI - Command Line Interface

How to Run the Kiuwan Local Analyzer via Command Line Interface

Command Line Interface (CLI)

Kiuwan Local Analyzer (KLA) can be run via C*ommand Line Interface* (*CLI*).

It works with Linux, UNIX, MacOS and Windows.

You need to choose the correct command line script :

- *$(AGENT_HOME)/**bin/agent.sh*** (Unix-like and MacOs)
- *$(AGENT_HOME)/**bin/agent.cmd*** (Windows)

Usage:

```
 <Launch script for your OS> (agent.cmd or agent.sh) [options]
param=val param2=val2 ...
```

Available *[options]* when running KLA in CLI mode are shown by typing :

```
agent.cmd -h (or --help)
```

Note: In UNIX systems, please check agent.sh has proper u+x permission attributes

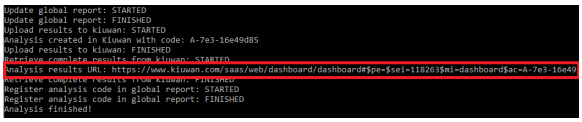
Local Analyzer Return Codes

After execution in CLI mode, KLA produces the following information:

- *URL to results of analysis*
- ***Return code*** (that can be used when using KLA in a script)

Regarding results of analysis, KLA will display (in stdout) the following information:

- Baseline scope
    - URL to Kiuwan dashboard with analysis results
- Delivery scope
    - URL to audit execution results



Kiuwan Local Analyzer command scripts (agent.cmd and agent.sh) return **0** after a *successful execution*.

- In Windows you can check return code in %ERRORLEVEL%
- In UNIX you can check it in $?

You can find detailed information on non-zero (error) return codes at Local Analyzer Return Codes

Synchronous and Asynchronous execution (-wr option)

A complete Kiuwan analysis involves two phases:

- *1st Phase - Local Analysis*
    - KLA analyzes source files and upload reports to Kiuwan
- *2nd Phase - Cloud Metrics Calculation*
    - Indicators and metrics are calculated in the cloud based on uploaded analysis reports
    - Deliveries promotion to baseline

You can decide whether KLA should wait (or not) to finish the 2nd phase before returning to your invoking script or shell command by ***–wr*** option.

- If –wr is not specified (default mode), KLA finishes after 1st Phase (local execution of analysis), returning control to invoking shells/script.
- If –wr is specified, KLA will wait for the 2nd Phase (cloud calculus and/or promotion deliveries to baseline) to finish before returning control.

```
 -wr, --wait-for-results
 Wait for kiuwan to return complete results once the local analysis has
finished
 Default: false
```

Return codes depending on –wr and analysis scope (baseline or delivery) are explained in Local Analyzer Return Codes

Authentication through CLI

When using CLI mode, you can either specify your kiuwan credentials or let KLA read previously stored credentials.

If you use the local analyzer *GUI* (*$AGENT_HOME/kiuwan.cmd*), it will ask for credentials (user and password).

If the credentials are valid, KLA will encrypt and save these credentials within agent properties file (*$AGENT_HOME/conf/agent.properties*) and will be used in subsequent executions unless you change them.

Please remember that if you change the password for your username in the Kiuwan application, you need to change it in the agent as well.

In case you need to change (and encrypt) the credentials, you can specify:

```
-e, --encrypt-password
Encrypts kiuwan password
Default: false
```

Above option will ask you for the credentials and will store them encrypted.

You can also specify your credentials by using next options:

```
--user
Kiuwan username (overrides username value in agent.properties)
--pass
Kiuwan password (overrides password value in agent.properties)
```

These options can be used together with any other CLI option.

Basic execution of a baseline analysis

The simplest usage of KLA is to execute a *baseline analysis*.

To do it, just specify next options:

```
-s, --sourcePath
Directory with code to analyze

-n, --softwareName
Name of the target application

-c, --create
Create software at kiuwan service if not exists

-l, --label
Label for the analysis

-m, --model-name
The model name to use when analyzing
```

- Option ***–c*** will create the application in Kiuwan if it already does not exist (if the application already exists, -c will not have any effect).
- Option ***–s*** will specify the source directory where the source files are located.
- Option ***–l*** allows to specify a label to identify the analysis.
- If the application does not exist, CQM will be assigned as the model of application, unless you specify another model with **–m** option. Option –m allows specifying a quality model to use in the analysis, regardless the model associated to the application. Once used, the specified model will be associated to the app.

The most basic use of KLA in CLI mode is to run an analysis specifying source code path and application name.

This can be done by executing as below:

```
agent.cmd -s <root directory of source files> -n <application name> -c
```

Execution of a delivery analysis

The ***scope of an analysis*** may be one of the following.

- ***Baseline*** (development milestones are usually analyzed as baselines)
- ***Complete delivery*** (an important delivery of the whole application)
- ***Partial delivery*** (a delivery usually associated to a change request)

You can specify the scope of an analysis by using ***–as*** option:

```
-as, --analysis-scope
The analysis scope. One of [baseline|completeDelivery|partialDelivery].
Defaults to baseline.
```

Additional options are applicable to delivery analysis, such as:

```
-cr, --change-request
The change request associated with the delivery to analyze or promote
(implies -as completeDelivery if no other option is specified)

-crs, --change-request-status
The change request status. One of [inprogress|resolved]. Defaults to
resolved. Only applies to delivery scopes

-bn, --branch-name
The branch name associated with the delivery to analyze (implies -as
completeDelivery if no other option is specified)
```

Example:

```
agent.cmd -n <my_app_name> -s <my_source_dir> -l "My analysis label" -as
completeDelivery -cr "My change request" -crs resolved -bn "my branch name"
```

Delivery promotion to Baseline

Deliveries can be "*promoted*" to baseline without the need to run a new analysis.

To promote a delivery to a baseline means that a new baseline will be calculated based on delivery analysis results.

This new baseline will be calculated differently depending on the delivery scope:

- If the delivery is *Complete*, new baseline indicators will be calculated based completely on delivery defects, i.e. not taking into account any previous baseline analysis.
- If the delivery is ***Partial***, new baseline indicators will be incrementally calculated as follows:
    - For any new file contained in the delivery (i.e. not existing in the baseline), the file and associated defects will be included as part of the new baseline.
    - For every file contained in the delivery and in the baseline, defects found during the delivery analysis will overwrite defects found when analyzed in the baseline.
    - For those files contained in the baseline but not included in the delivery, previous defects will be maintained.

A delivery can be promoted to baseline specifying **--promote-to-baseline** option:

```
--promote-to-baseline
Promotes a single delivery to baseline (-n, -l and -cr must be specified)
or all pending deliveries (-n must be specified)
Default: false
```

When **--promote-to-baseline** option is specified, Kiuwan Local Analyzer will not run the delivery analysis, it will only "promote" the analysis results to the baseline.

Therefore, this option must be only executed on a previously analyzed delivery.

To identify the delivery to promote, you need to fully specify the delivery by indicating application name, change request and delivery label. In case that more than one delivery matches those values, most recent delivery will be selected to be promoted.

As an example, let's promote a delivery of the application "My application" that was created with change request "CR-9999" and label "My delivery label". In this case, we want to label the new delivery as "New baseline label".

The command to execute will be:

```
agent.cmd --promote-to-baseline -n "My application" -cr CR-9999 -l "My
delivery label" -pbl "New baseline label"
```

Supplied parameters are:

- --promote-to-baseline: the action will be promotion to baseline (not delivery analysis)
- --softwareName (-n): name of the application the delivery belongs to
- --label (-l): label of the delivery to be promoted
- --change-request (-cr): change request of the delivery to be promoted
- --promote-to-baseline-label (-pbl): label for the new baseline

> Important:
>
> -n, -l and -cr options are mandatory (-l and -cr options admit empty strings)
>
> -pbl is optional (if not specified, the new baseline will have the same label as the promoted delivery)

After execution, you will get next message indicating the promotion has been successfully done:

```
Running delivery promotion mode...
 application = My application
 delivery label = My delivery label
 change request = CR-9999
 baseline label = New baseline label
 Delivery promoted. New baseline analysis code = A-7e0-1577a12f767
 Analysis results URL: <<link to Kiuwan page with results of just created
application baseline>>
 Promotion done!
```

Wait for promote

By default, execution will finish without waiting for the promotion process to be finished. This means that the promotion process will still be running in the cloud but control is returned immediately.

If you need to wait for the promotion to be finished, you should specify *-wr* option.

Bulk mode promotions

> Kiuwan Local Analyzer allows *to promote a group of deliveries in a single step* (*bulk mode*).
>
> All pending deliveries of the selected application with *'Accepted for promotion'* enabled will be promoted.
>
> Command to execute will be:
>
> agent.cmd --promote-to-baseline -n "My application"

As with single promotions, you can specify -wr option to wait for the promotion to finish,

Please visit Deliveries Promotion to Baseline for further details.

Log files, temporary analyses directories and lock files

During analysis execution, Kiuwan Local Analyzer generates *working temporary log files and directories* u nder ***$(AGENT_HOME)/*temp** directory:

- agentGUI.log
- agent.log
- analyses' log directories (named as appname.serial_number)
- models cache directory (models directory)
- *.lock files (used for automatic upgrade synchronization in parallel execution environments)

These files/dirs are generated by Kiuwan Local Analyzer and, in case of error, you can browse them or submit to Kiuwan Technical Support.

If GUI is used, those files will be removed after analysis execution, but if CLI is used those files will remain after analysis.

You can clean those temporary files using the following option:

```
--clean

     Remove lock files and temporary analysis directories

     Default: false
```

Alternatively, you can configure Local Analyzer to automatically remove them after CLI analysis execution.

- To do it, you must set *clean.results.data=true* in *$(AGENT_HOME)/conf/agent.properties* (defaul t value: false).

There's an exception to above: agent.log. This file remains and is not deleted under different GUI and CLI invocations.

By default, agent.log grows up to 64 Mb of size. When it reaches that size, file is rolled up and contents are backed up to another file (with an index suffix). This can happens up to 5 times.

This default behavior is defined at *$(AGENT_HOME)/conf/log4j.properties* but you can change it to suit your specific needs.

KLA Configuration

Configuration of KLA is quite flexible, and it's been designed to allow a fine-grain configuration of the analyses.

Although there are many configuration options, most commonly customized are:

- *encoding* : encoding used for source files parsing (default: UTF-8)
- *analysis timeout* : max duration allowed for a single analysis task (default: 60 min)
- *max memory* : max amount of memory allowed for a single analysis task (default: 1 Gb)
- *files exclusion pattern*: set of files to exclude from analysis

As said before, bear in mind that *GUI is just a visual wrapper of command line KLA,* so, regardless you use GUI or CLI, your configuration of KLA keeps uniquely maintained in configuration files that are read by both modes, GUI and CLI. When using GUI you may not find all the available parameters. GUI only allows to modify the most important ones. If you don't find it go to the configurations files.

> See ***$AGENT_HOME/conf/analyzer.properties*** for a full listing and explanation of available configuration parameters.

Configuration scopes and precedence

> KLA configuration is quite flexible and it's been designed to allow configuring the analysis at different scopes:
>
> - *installation-wide*,
> - *application-specific*, and
> - *single-analysis*

For example, default encoding for source files parsing is set to UTF-8. But you might want to maintain that default encoding and modify it only for a specific application or even for a specific analysis.

KLA provide means to configure at the level you need. But *it's quite important to know the precedence of every configuration and how it's applied.* thus avoiding headaches (...)

Next table shows the different scopes and precedence among them..

You must understand it as: *a more specific configuration always takes precedence (mandates) over a more general one*.

Precedence	Scope	Description	Configuration
1st	Analysis	Configuration applies to a unique analysis	Properties specified directly in command line (property=value or .kiuwan.property=value) Properties specified in .kiuwan file
2nd	Application	Configuration applies to all the analyses of a specific application	Application-specific configuration file ($AGENT_HOME/conf/apps/<appname>.properties)
3rd	Installation	Configuration applies to all the analyses of a KLA installation	Installation-wide configuration file ($AGENT_HOME/conf/analyzer.properties)

Installation-wide and Application-specific configuration

Both installation-wide and application-specific configurations are done by setting properties in specifics ***analyzer.properties*** files.

- *global configuration file*: ***$AGENT_HOME/conf/analyzer.properties***
- *application-specific configuration file*: ***$AGENT_HOME/conf/apps/<appname>.properties***

Any property configured in the global config file will apply to all the analysis unless there is an application-specific config file. If exists, application-specific configuration will be applied instead of general one.

You can directly change configuration properties in global config file and even create an app-specific config file. Within an app config file, you can include any general configuration property and set its value to your needs.

If you use GUI, config files are automatically updated as you modify existing values, asking you to decide if a configuration must be globally applied or only for a specific application.

Insights configuration

Insights configuration can only be customized installation-wide.

To do so, edit the Insights configuration file located at $AGENT_HOME/conf/insight.properties

Here you can configure:

- Custom Maven repositories: please refer to Insights - Additional Maven repositories documentation page for more details about these options.
- Gradle version to use for gradle build files: this will be the Gradle compatibility version that Insights analyzer will use when inspecting gradle build files. You can see all available versions at https://services.gradle.org/distributions/. By default this property is set to "6.2.1". Please note that this means that existing analyses may have used the previously supported Gradle vesion (4.7).

Configuration at Analysis-time (.kiuwan file)

Besides the above available options to configure the analysis, you can also configure some properties at analysis-time, i.e. at the moment of analyzing your source code.

The mechanism to use is based on setting properties at ***.kiuwan*** *configuration file*.

Through this possibility, you can modify “static” properties such as:

- The Model to be used in the analysis
- The Portfolio value of the analysis
- The Audit to be applied to the analysis

By “static” we mean properties that are defined mainly at application level.

These analysis-time modifications are mainly used when Kiuwan is integrated within a fully-automated Continuous Integration System.

This option allows automatically creating and configuring applications during the analysis, without the need to do it by manually configuring your Kiuwan website. This allows delegating the creation of applications, portfolios, etc. to the Continuous Integration system.

.kiuwan configuration format

The ***.kiuwan*** configuration file must meet these requirements:

- It must be placed in the root of the source folder directory
- It must be named ".kiuwan"
- It must be UTF-8 encoded

By including this file in your base source folder you can configure the following characteristics in the analyzed application

- *Portfolios* classification. Set the application classification under your custom portfolios and built-in portfolios ("Business Value" and "Provider").
- Model to use in the analysis. Set the model to use when analyzing the application.
- *Audit* to be applied to the analysis results.
- Source file *encoding*. Set the encoding of the source files under analysis.
- *Includes* pattern. Set the files to include in the analysis (a comma separated list of ant patterns).
- *Excludes* pattern. Set the files to exclude from the analysis (a comma separated list of ant patterns).

(*) IMPORTANT: See Persistence of changes to understand how this setting affects depending on the Analysis Scope (i.e. if it's applied during a baseline or a delivery analysis).

It must follow this format:

```
application.businessValue=(CRITICAL| HIGH | MEDIUM | LOW | VERY LOW)
application.provider={provider value}
application.portfolio.{portfolio 1 name}={portfolio 1 value}
application.portfolio.{portfolio 2 name}={portfolio 2 value}

(analysis.model | analysis.model.kiuwan)={model name}
analysis.encoding={source files encoding}
analysis.includesPattern={includes pattern}
analysis.excludesPattern={excludes pattern}
```

Available parameter values are:

Property	Parameter	Meaning
Business Value portfolio classification	application.businessValue= (CRITICAL | HIGH | MEDIUM | LOW | VERY LOW)	Sets the Business Value*
Provider portfolio classification	application.provider={provider value}	Sets the Provider*
Custom portfolio classification	application.portfolio.{portfolio 1 name} ={portfolio 1 value}	Sets custom portfolio values
Model	analysis.model={model name}	Sets the model to use when analyzing the application
Force model in deliveries	analysis.forceModelInDeliveries=true	When set, deliveries will be analyzed using the last version of the application's model
Audit	analysis.audit={audit name}	Sets the Audit to be applied to the analysis results
Source files encoding	analysis.encoding={source files encoding}	Sets the encoding of the source files under analysis
Includes pattern	analysis.includesPattern={includes pattern}	Set the files to include in the analysis (a comma separated list of ANT patterns).

Excludes pattern	analysis.excludesPattern={excludes pattern}	Set the files to exclude from the analysis (a comma separated list of ANT patterns).
Additional analysis notification address	analysis.notification={email address}	Email address to send analysis notification

For Kiwan Local Analyzer command-line scripts, the new arguments are --argsFile (or -@ in short form), --bomFile (or -b for short), and --bomFormat.

The following arguments for the command-line interfaces are now available:

- **argsFile:** Arguments file path, encoding arguments, one per line. Useful for sharing common arguments. Explicit arguments overwrite the ones in the args file.
- **bomFile:** Bill-of-materials file path. Lists files to process with a certain format.
- **bomFormat:** Bill-of-materials format. Currently two formats are supported: 'raw' (one file path per line, absolute or relative to analysis directories), and 'json-comp-db' (JSON Compilation Database, a JSON array of command objects specifying each source unit, and how it is compiled).

Most commonly configured parameters

Below you can find how to change default parameters for *most common situations*.

All those configuration parameters are globally specified at ***$AGENT_HOME/conf/analyzer.properties*** (or at ***$AGENT_HOME/conf/apps/<appname>.properties*** for a specific application).

Encoding

When KLA analyzes source files, it uses ***UTF-8*** *as default* encoding.

If you are using locale-specific characters in your source code you may need to change this default encoding to another one that is more adequate to your language.

If you see a high number of unparsed files most probably you need to change default encoding in *analyzer.properties* file.

```
# Default encoding for source files
encoding=UTF-8
```

Memory

For every task during analysis, KLA starts a JVM to perform the required analysis step.

KLA configure the max amount of memory to be used during the analysis to ***1GB*** *as default*.

Depending on the source code, 1 GB might not be enough and you should increase it. To do that change below property:

```
# Maximum size for heap memory (1024m = 1 Gigabyte)
memory.max=1024m
```

Timeout

KLA is configured for a maximum duration of a single analysis step (***60 min*** *as default*).

Due to code size, that time might not be enough to finish the analysis and a timeout error appears.

In that case you can increase max duration by setting below property :

```
# Timeout to use for max execution time of each analysis step
timeout=36000004m
```

Please note that duration is set in *milliseconds*. Through the GUI, the value is in minutes (GUI will convert it to milliseconds when updating config file).

Supported Technologies

It's important to mention a "special" configuration parameter: "*supported.technologies*".

This parameter specifies the ***current technologies supported by Kiuwan Local Analyzer***.

```
# Current technologies supported by agent
supported.technologies=abap,aspnet,c,cobol,cpp,csharp,html (etc..)
```

Source files and technologies are "associated" through file extensions. And this association is used by KLA to execute the adequate engine on the source files.

But there are some extensions that are commonly linked to more that one technology:

- .sql is a typical example, it matches plsql, transact and Informix,
- .c/.h are also the case for c, cpp and objective-c

When running in GUI mode, KLA detects such ambiguous situations and asks the user to resolve it by selecting the adequate technology.

Then, the user might select plsql because he/she knows that it's analyzing an oracle application.

When running in CLI mode, if none is specified, KLA will run (in the sql case) the three available sql engines, wasting time and resources and producing confusing results (as will generate defect information from all those engines).

An easy way to avoid unnecessary processing is specifying *supported.technologies* parameter with only the proper technologies when invoking KLA in CLI mode.

In this case, transact and informix should be deleted from the list of supported technologies.

As a general rule, only the indicated engines specified in *supported.technologies* parameter will be executed in the analysis.

For further info please visit Kiuwan Supported Technologies

Portfolio values, Model and Audit

You can configure Portfolios, Model and Audit through several means:

- By using the **Kiuwan Website** (*Application Management section*)
- By using the **Graphical User Interface** (GUI) of KLA (GUI - Specification of Model and Portfolio at Analysis-time)
- By using the **Command Line Interface** (CLI) of KLA and .**kiuwan configuration** file.

The table below shows how to configure these properties through **CLI options** and **.kiuwan configuration** file:

Property	CLI option	.kiuwan configuration file
model	-m, --model-name The model name to use when analyzing	analysis.model={model name}
forceModelInDeliveries	.kiuwan.analysis.forceModelInDeliveries=true	analysis.forceModelInDeliveries=true
audit	.kiuwan.analysis.audit={audit name}	analysis.audit={audit name}
businessValue portfolio	.kiuwan.application.businessValue=(CRITICAL | HIGH | MEDIUM | LOW | VERY LOW)	application.businessValue=(CRITICAL | HIGH | MEDIUM | LOW | VERY LOW)
provider portfolio	.kiuwan.application.provider={provider value}	application.provider={provider value}
"custom" portfolio	.kiuwan.application.portfolio.{portfolio 1 name}={portfolio 1 value}	application.portfolio.{portfolio 1 name}={portfolio 1 value}
additional email address notification	.kiuwan.analysis.notification={email address}	analysis.notification={email address}

Persistence of changes depending on Analysis Scope (baseline or delivery)

Regardless the way you use (GUI, CLI or .kiuwan file), **persistence of configuration changes depends on Analysis scope.**

- *Baseline*
  - Changes to portfolios, model, forceModelInDeliveries, and audit will be *used in the analysis*, *applied to the Application*, and *will remain persistent for future analysis* (i.e. *future analyses will use those values*)
- *Delivery* (Complete or Partial):
  - Portfolio values, model and audit will ONLY be applied to the current analysis, remaining unchanged application's values (i.e. *further analyses will use application's values*)
    - Note: model will be applied only if the application is configured to Force Model In Deliveries
  - forceModelInDeliveries is ignored when used in delivery analyses

Using third party analyzers

Kiuwan Local Analyzer can execute third party analyzers as well as importing results from 3rd-party analyzers.

Please visit Third party analyzers for further info.

Troubleshooting

If you are experiencing problems with KLA execution please visit Troubleshooting