# Structure

The structure of an Odysseus Script may contain different things: commands, comments, variables, constants, macros or control flows.

## Commands

Commands are normally those statements that are send to Odysseus, e.g. to install a query or to configure a setting. Each command in Odysseus Script begins with a hash/number sign (#) followed by its name and by some parameters (if the command needs some parameters).

```
#COMMAND parameter1 parameter2
```

Normally, one command is executed for its own and has no impact to other commands. However, the #QUERY command needs current settings like the parser, which is set by the command #PARSER.

```
#PARSER PQL
...
```

**Remark: It is important to define the right parser as Odysseus cannot determine the parser based on the query text!**

## Comments

Comments mark lines that should be ignored by the parser. Useful for additional information for the reader. Comments are defined by using three slashes per line. Currently, there is no way to comment multiple lines at once.

```
///this is ignored by the parser
```

## Variables

Variables can be used to reuse certain values. In most cases, they are moved to the top of a file so that they become more visible. A variable can be created by using #DEFINE and be removed by using #UNDEF. If a variable should be calculated from other variables (or constants), #EVAL can be used. To access a variable, the user has to write `${....}`. The existence of a variable (if it is created or not) can be checked with #IFDEF. The value of a variable can be printed by using #PRINT (into the console). The following example shows three variables: an integer called "currentid" that has the value "50", a variable named "path", which has the value "F:/odysseus/example/" and a calculated variable "maxid".

```
#DEFINE currentid 50
#DEFINE path F:/odysseus/example/
#EVAL maxid = currentid + 100
#RUNQUERY
SELECT * FROM example WHERE id >= ${currentid} AND id <= ${maxid}
#RUNQUERY
CREATE STREAM source (id Double, data STRING)
    WRAPPER 'GenericPush'
    PROTOCOL 'CSV'
    TRANSPORT 'File'
    DATAHANDLER 'Tuple'
    OPTIONS ( 'filename' '${path}input.csv')
```

The first and third variables "currentid" and "maxid" are used in the first query, so that it is equal to "SELECT * FROM example WHERE id >= 50 AND id <= 150". The second variable is used in the second query as a prefix for the filename. Note that variables are simply replaced by its values.

## Constants

Constants are special varibales that exists without defining them explicitly. For example, a default variable is NOW so that ${NOW} can be used to get the current time in milliseconds. This is might be useful if the time of the script execution is needed (e.g. for filenames). The following table shows an excerpt of currently available constants. Developers can add application-specific constants, if needed (see Additional constant variables).

| Symbol | Value |
|---|---|
| NOW | Current timestamp |
| WORKSPACE* | The absolute path to the workspace |
| PROJECT* | The project name |

| | |
|---|---|
| PROJECTPATH* | The absolute path to the project |
| BUNDLE-ROOT | same as PROJECPATH but can additionally be used in tests and Autostart |
| WORKSPACEPROJECT* | The absolute path to the workspace extended by the project name |
| ROOT* | The absolute path to the current file |
| OS.ARCH | The operating system architecture |
| OS.VERSION | The operating system version |
| OS.NAME | The operating system name |
| CPU | The amount of available processors |
| MEM | The total amount of memory |
| VM.NAME | The name of the Java VM |
| VM.VENDOR | The vendor of the Java VM |
| VM.VERSION | The version of the Java VM |

*Only available in Odysseus Studio

Additionally, system properties / environment variables provided by System.getProperty(...) and `System.getenv(...)` are also available. To avoid naming collisions, each system property has "_" as prefix. Example: ${_user.name} will be replaced with the current system user name (NOT the username in Odysseus).

## Control Flows

Control flows are statements that are used to define which commands are executed, which not and how often they are executed. There are simple control flows like a for-loop (#LOOP) or a if-then-else (#IFDEF). They are explained in more details below.

## Procedures and Macros

Procedures and macros gives the user a possibility to reuse a certain snippet of the code. They can be distinguished between parameterizable procedures and simply reusable macros. Another advantage: Procedures (#PROCEDURE) are stored in the data dictionary so that their availablity is (according to the user's rights) system wide.