Other Commands

#CONFIG

Allows to define additional processing informations, e.g. IsSecurtyAware.

Example

```
#CONFIG isSecurityAware true
```

#DEFINE

This command is used to define variables to reuse certain values. See also at Variables how to use a defined variable or at #IFDEF to see how to use defined variables within if-statements.

Parameters

The parameters are one or two values, which are separated by a blank or tab. Therefore, the parameters may not have any blanks or tabs. The first parameter is the name and the second paramter is the value that is assigned to the name. However, the second parameter (the value) is optional, because if a value for #IFDEF is not necessary, otherwise it cannot be used for replacement.

Example

The first variable is called "one" and has no value. The second variable is called "two" and has the value "1234". See at Variables or at #IFDEF for examples how to use a variable.

```
#DEFINE one
#DEFINE two 1234
```

#EVAL

It behaves like #DEFINE, except it evaluates the expression to a result, which is then stored in the specified variable.

Parameters

The parameters are two values, which are separated by a blank, tab or "=". Therefore, the parameters may not have any blanks or tabs. The first parameter is the name and the second paramter is the expression, whose value is assigned to the name.

Example

The first variable is called "aNumber" and is a normal #DEFINE. "bNumber" gets the evaluated result of "aNumber + 2000": "3000". "cNumber" demonstrates an alternative way to specify new variables (omitting "=").

```
#DEFINE aNumber 1000
#EVAL bNumber = aNumber + 2000
#EVAL cNumber aNumber + 2000
```

Remarks:

- If you want to access elements that are defined (e.g. with another DEFINE) you will need to omit the "${}"
- You cannot use a loop variable here

#INPUT/#INCLUDE

This command copies the input from the source given into the current script file. The source can be a local file or a file on a web server (giving an URI).

```
#INPUT ${WORKSPACEPROJECT/}Source.qry
#INCLUDE http://odysseus.offis.uni-oldenburg.de/download/test/StreamSources.qry
```

#MDASTORE_DROP

This command drops an existing MDA store. See MDA store functions.

Parameters

The name for the store.

Example

```
#MDASTORE_DROP MyStore
```

#MDASTORE_INIT

This command creates a new MDA store. See MDA store functions.

Parameters

1. The name for the store
2. A list of the dimensions for the MDA store. Each dimension must contain the following information separated by colons:
    a. The lower border for the dimension
    b. The upper border for the dimension
    c. The number of borders for the dimension (incl. lower and upper)

Example

```
#MDASTORE_INIT MyStore 0:200:3 100:300:5
```

#ODYSSEUS_PARAM

Can be used to set internal Odysseus configuration params. This should be only

Parameters

The parameters are: The name of the Odysseus configuration param and the new value

Example

```
#ODYSSEUS_PARAM scheduler_TimeSlicePerStrategy 10
```

#PRINT

For debugging purpose the values of variables that are defined using #DEFINE or arbitrary expressions can be printed to the std output using #PRINT.

```
#DEFINE path F:/odysseus/example/
#PRINT path
#PRINT "Running on "+toString(OS.NAME)
```

#RELOADFROMLOG

The reload log is a file that logs all queries that were sucessfully installed into the system. This command can be used to run these logged queries from the log, e.g. to recreat an old ystem state.

Parameters

This command has no parameters.

Example

```
#RELOADFROMLOG
```

#SCHEDULER

Sets the used scheduler and its scheduling strategy.

Parameters

It needs two parameters: The scheduler and the scheduling-strategy. The available schedulers and strategies depends on the current system setting (additional features could be necessary!), because they are dynamically bound.

Example

Uses the "Single Thread Scheduler" with a "Round Robin" scheduling strategy

```
#SCHEDULER "Single Thread Scheduler RR" "Round Robin"
```

Uses the "Single Thread Scheduler" with a "Aurora Min Cost" scheduling strategy

```
#SCHEDULER "Single Thread Scheduler RR" "Aurora Min Cost"
```

Uses the "Single Thread Scheduler" with a "Aurora Min Latency" scheduling strategy

```
#SCHEDULER "Single Thread Scheduler RR" "Aurora Min Latency"
```

Uses the "Single Thread Scheduler" with a "Chain" scheduling strategy

```
#SCHEDULER "Single Thread Scheduler RR" "Chain"
```

Uses the "Single Thread Scheduler" with a "Biggest Queue" scheduling strategy

```
#SCHEDULER "Single Thread Scheduler RR" "Biggest Queue"
```

Uses the "Simple Dynamic Priority Scheduler" with a "Round Robin" scheduling strategy

```
#SCHEDULER "Simple Dynamic Priority  Scheduler" "Round Robin"
```

#SLEEP

This command can be used to wait a certain time before executing the next command

Parameters

The parameter is a number. It defines the time in milliseconds for which the script execution should sleep.

Example

Waiting 2 seconds (2000 milliseconds) until the next command is invoked.

```
#SLEEP 2000
```

#STARTSCHEDULER

This command starts the scheduling.Notice that the scheduling strongly influences the processing and should be carefully used. The scheduler is running by default. You can stop it by using #STOPSCHEDULER

Parameters

This command has no parameters.

Example

```
#STARTSCHEDULER
```

#STOPSCHEDULER

This command stops the scheduling.Notice that the scheduling strongly influences the processing and should be carefully used. The scheduler is running by default. You can start it by using #STARTSCHEDULER

Parameters

This command has no parameters.

Example

```
#STOPSCHEDULER
```

#UNDEF

With this command, a given value for a variable can be removed and a new value can be assigned with #DEFINE.

Example

```
#UNDEF variable
```