Key Value Store Feature

Odysseus provides a simply feature for storing, retrieving and removing of values from a key value store.

Out of the box, odysseus provides

- MemoryStore: A key value store that keep the values in memory only
- FileStore: A key value store that keeps the data in main memory and stores it additionally to disk
- With a plugin you could write to a Redis-Store.

Default Store

For each user, there exisists a default store. This is a memory store and has no name. To access this store, left parameter for store name empty. This store is created, when the store is first accessed.

Creating a store

A store can be created with the Odysseus Script command: #CREATE_KV_STORE name type (PARAM_1_KEY=PARAM_1_VALUE,..., PARAM_N_KEY=PARAM_N_VALUE)

To create a Memory Store simply use:

```
#CREATE_KV_STORE test MemoryStore
```

Some stores have parameters, e.g. the FileStore (with the paramter filename):

```
#CREATE_KV_STORE test FileStore (filename=c:/temp/tmp.store)
```

For all stores, the normal behavior is to allow creation only if there exists no store with the desired name. But with the parameter "createOnlyIfNotExists", one can change that behavior. If set to false, a store with the same name is not droped and the existing store will be reused (Warning, in this case the type of the store cannot be changed! So be careful with this option!):

```
#CREATE_KV_STORE test MemoryStore(createOnlyIfNotExists=false) /// default is true
```

Access a store

MEP functions can be used to read an write from the store. So this can be used in (mostly) all operators that contain expressions or predicates (e.g. Map operator, Select operator, Route operator)

- kvwrite(String StoreName, String KeyName, Object Value): (Overwrites) a value in the give store and key
- kvread(String StoreName, String KeyName): reads the value from the store with the key
- kvremove(String StoreName, String KeyName): remove the value from the store

The default store is used, when the storeName is not given.

The following code block shows an example

```
#PARSER PQL
#RUNQUERY
timer = TIMER({
            period = 1000,
            source = 'timersource'
          }
        )

Map1 = MAP({EXPRESSIONS = ['time','kvwrite("test2","Time",time)','kvread("test2", "Time")']}, timer)
```

For easier handling, there are also operators that allow to read different datatypes from store and avoid casting in MAP, as the default return value of the kvread method is Object:

- kvReadBool
- kvReadLong
- kvReadDouble
- kvReadString

Important: This read operations make no type conversion! So if you call kvReadBool and the value is a String, there could be Problems when using this value later.

You could also use an operator to write to an existing store:

StoreWriter

```
#PARSER PQL
#RUNQUERY
timer = TIMER({
            period = 1000,
            source = 'timersource'
          }
        )

out = StoreWriter({IdAttribute='time', store="test2")},timer)
```

OdysseusScript

Writing to and removing from a store can also be done with Odysseus script. Remark that these functions will only be executed once to a script. If a query is stopped and started again this will not call the Odysseus script part again

```
#KV_STORE_WRITE STORENAME KEY VALUE
#KV_STORE_REMOVE STORENAME KEY
```

As in other cases, when omitting the STORENAME the users default store is used.

For special cases, there is a new function, that can read a string from file into a store:

```
#CREATE_KV_STORE query_store MemoryStore (createOnlyIfNotExists=false)
#KV_STORE_WRITE_FROM_FILE query_store 0 ${PROJECTPATH}/SimpleQuery_0.qry
#KV_STORE_WRITE_FROM_FILE query_store 1 ${PROJECTPATH}/SimpleQuery_1.qry
```