KPI_Low: installation

1. **Export global variables**

   export PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/home/{USER}/inst/lib/pkgconfig
   export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/{USER}/inst/lib

2. **Install scew**

   - Download scew (scew-1.1.3.tar.gz)
     http://nongnu.askapache.com/scew/scew-1.1.3.tar.gz
   - Unpack
     tar -zxf scew-1.1.3.tar.gz
   - cd scew-1.1.3
   - Install

     ./configure --prefix=/home/{USER}/inst
     make
     make install

KPI_Low: installation

3. **Install KPI_Low**

   - Download KPI_Low (KPI_low.tar.gz)
     http://cs.karelia.ru/
   - Unpack
     tar -zxf KPI_low.tar.gz
   - cd KPI_low
   - Install

```
./autogen.sh
./configure --prefix=/home/{USER}/inst
make
make install
```

KPI_Low: example

4. **Example**

   - Download example (kpilow_example.tar.gz)
     http://cs.karelia.ru/
   - Unpack
     tar -zxf kpilow_example.tar.gz
   - cd kpilow_example
   - Edit Makefile

     gcc -Wall -g -pthread -lkpilow -I/home/{USER}/inst/include/ -L/home/{USER}/inst/lib -o consumer_kp consumer_kp.c
     gcc -Wall -g -pthread -lkpilow -I/home/{USER}/inst/include/ -L/home/{USER}/inst/lib -o publisher_kp publisher_kp.c

KPI_Low

1. **API**

   <kpi_low.h>

   ...

2. **Basic scenarios**

   - Initialize connection to Smart Space
   - Send set of triplets to Smart Space
   - Receive set of triplets from Smart Space
   - Subscription to triplets
   - Asynchronous subscription

KPI_Low: Initialize connection

```
ss_info_t ss_info;

/* Discovering new Smart Spaces */
ss_discovery(&ss_info);

if (ss_join(&ss_info, "KP name") == -1)
{
    /* Error handle */
}

/*
... KP logic ...
                    */

ss_leave(&ss_info);
```

KPI_Low: Send set of triplets

```
ss_triple_t * triple = NULL;

/* insert sensor to the smart space */
ss_add_triple  (  &triple, ss_info->space_id, "hasSensor", Name,
                  SS_RDF_TYPE_URI, SS_RDF_TYPE_BNODE);
ss_insert(ss_info, triple, bnodes);
ss_delete_triples(triple);
triple = NULL;

/* add sensor value to the smart space */
ss_add_triple  (  &triple, bnodes->uri, "hasName", Name,
                  SS_RDF_TYPE_URI, SS_RDF_TYPE_LIT);

ss_insert(ss_info, triple, NULL);
ss_delete_triples(triple);
```

KPI_Low: Receive set of triplets

```
ss_triple_t * sensor_info_rqst = NULL;
ss_triple_t * result_triple = NULL;

/* query for triplets */
ss_add_triple  (  &sensor_info_rqst, sensor->uri, "hasName",
                  SS_RDF_SIB_ANY,
                  SS_RDF_TYPE_URI, SS_RDF_TYPE_URI);
if(ss_query(ss_info, sensor_info_rqst, &result_triple) < 0)
{ /* Unable to query */ }

ss_delete_triples(sensor_info_rqst);

/* ... Handle result_triple ... */

ss_delete_triples(result_triple);
```

KPI_Low: Subscription to triplets

```
ss_triple_t * triple_rqst = NULL;        ss_triple_t * n_val = NULL;
ss_triple_t * triple = NULL;             ss_triple_t * o_val = NULL;

/* query for triplets */
ss_add_triple (&triple_rqst, sensor->uri, "hasValue",
                    SS_RDF_SIB_ANY, SS_RDF_TYPE_URI, SS_RDF_TYPE_URI);

if(ss_subscribe(ss_info, subs_info, triple_rqst, &triple) < 0)
{ /* Failed to subscribe */ }
ss_delete_triples(triple_rqst);

int status = ss_subscribe_indication(&ss_info, subs_info, &n_val,
&o_val, 3000);

if(status == 0) continue; /* timeout */
if(status < 0) { /* Error occurred */ }
if(status == 1) { /* … new values ... */ }
```

KPI_Low: Asynchronous subscription

```
pthread_t thread;
thread_param_t * param;

/* ... prepare to subscription and subscribe */

ss_add_triple (&triple_rqst, sensor->uri, "hasValue",
                    SS_RDF_SIB_ANY, SS_RDF_TYPE_URI, SS_RDF_TYPE_URI);

param = (thread_param_t *) malloc(sizeof(thread_param_t));

param->ss_info = *ss_info;
param->subs_info = subs_info;

if (pthread_create(&thread, NULL,
                                    subscribe_handler, (void *)param))
{ /* handle error */ }
```

KPI_Low: Hello World

Task for you

- Use **“Hello World”** triplets
- **First KP** should publish property “Hello world”
- **Second KP** should receive published property

Additional:

- Extend Second KP to **subscription**
- Extend Second KP to **asynchronous** subscription