HORSE ID BAYESIAN BELIEF NETWORK MODEL MANUAL

To query the HorseID BBN Server use bash script with curl is as follows:

```
curl -X  < dev_environment > {url_address} / < object > / < function > --data  < request >
```

This is equivalent to the Python code:

```
< object >.< function > ( < request > )
```

Where **< dev_environment >** can be any of:

GET	For Testing	for testing the function without a test case in testing env.
PUT	For Usage	for using function with a request in production or dev env.
POST	For Testing	for testing the function with a desired test case in test env.

Where **< object >** can be any of:

variables or model (We will focus on model for now)

Where **< function >** can be any of:

build	use_default_values	get_cpds	load_data	check_model
run	declare_variables	load_cpds	prepare_data	get_cardinality
update	update_values	draw_default_graph	train_model	get_local_independencies
initialise_space	load_sizes	draw_graph	update_model	get_active_trail_nodes
set_universe	set_evidences	build_model	test_model	query
clear_values	set_cpds	load_cpd_to_model	describe_data	map_query

The general **< request >** data format is :

```
request = {
        data :{
                'variable1':            values1,
                'variable2':            values2,
                ...
                'variableN':            valuesN
        },
        dataset:{
                'variable1':            [values1],
                'variable2':            [values2],
                ...
                'variableN':            [valuesN]
        },
        'graph': [
                (variable1_i,           variable1_j),
                (variable2_i,           variable2_j),
                ...
                (variableN_i,           variableN_j)
            ],
            'node'          :       ‘variable_symbol',
            ‘variables':{
                variable1: values,
                variable2: values,
                ...
                variableN: values
            },
            ‘variable_card':{
                variable1: values,
                variable2: values,
                ...
                variableN: values
            },
            'values'            :               [values],
            'observed'          :               'values',
            'evidence'          :               [values],
            'evidence_card'     :               [values],
            'elimination_order' :               [values]
        }
```

For example, to call function build (None) on the Horse Identification BBN model, that is Model.build(request=None), then call:

for using the function in development and production mode:

```
curl -X PUT http://locahost:8000/model/build --data  { ‘node’ : ‘value’ }
```

for testing the function without a test case in testing mode:

```
curl -X GET http://locahost:8000/model/build
```

for testing the function with a test case of { ‘node’ : ‘value’, ‘result’: True } in testing mode:

```
curl -X POST http://locahost:8000/model/build --data  { ‘node’ : ‘value’, ‘result’: True }
```

General work flow in the development and production environment is given by:

1. BUILD SYSTEM

    Python Code:
    ```
    from bbn import HorseIDBayesianNetwork
    bbn = HorseIDBayesianNetwork( );
    bbn.build( );
    ```

    Curl/REST API code:
    ```
    curl -X PUT http://localhost:8000/model/build  --data { }
    ```

2. RUN SYSTEM

    Python Code:
    ```
    bbn.run( );
    ```

    Curl/REST API code:
    ```
    curl -X PUT http://localhost:8000/model/run  --data { }
    ```

3. USE SYSTEM

    Python Code:
    ```
    bbn.set_cpds( request );
    ```

    Curl/REST API code:
    ```
    curl -X PUT http://localhost:8000/model/set_cpds --data $request
    ```

NOTE: Step 1 and Step 2 are very import to start up the BBN system. All other activities are done in Step 3.

Hence, we have the general procedure is as follows:
Python code:
```
from bbn import HorseIDBayesianNetwork;
bbn = HorseIDBayesianNetwork

#set up the system.
request=None
bbn.build(request)
bbn.run(request)

#activities here:
request = {…}
bbn.query(request);
```

Curl/REST API code:
```
#set up the system
curl -X PUT http://localhost:8000/model/build  --data { }
curl -X PUT http://localhost:8000/model/run  --data { }
#activities here:
request={ }
curl -X PUT http://localhost:8000/model/query --data $request
```