title: Create a world model weight: 15

Create a world model for John and his home. Create an agent that subscribes to changes in the world model.

Before you begin

1. Install GIT.
2. Install NodeJS v8.
3. Clone the Knowledge and Reasoning SDK.
4. Register for an IBM Cloud account.
5. Install the IBM Cloud CLI tool.
6. Install Python 2.7.

Procedure

1. In the `kr-node-sdk` folder, add .bak to the files `homeSecurity.js`, `condition.js`, and `action.js`. **Note**: You create these files during the tutorial. Use these files as a reference if your tutorial fails to run.
2. Create a home security app,`homeSecurity.js`, in the `kr-node-sdk` folder. Add the following code to the start of the file:

```
require('dotenv').config();
const express = require('express');
const bodyParser = require('body-parser');
const request = require('request');
const actions = require('./action');
const conditions = require('./condition');
const Agent = require('./sdk/messages');

const app = express();

app.use(bodyParser.json());

var KnowledgeObject = require('./sdk/object');
var KnowledgeRelation = require('./sdk/relation');
```

3. Write a function to create a person, a house, and a door object in local memory. Use the `KnowledgeObject` object.

```
// Create objects in local memory
var person = new KnowledgeObject('Person',
  {
    'name': 'John',
    "latitude": 12.456,
```

```
    "longitude": 67.99
  }
);

var house = new KnowledgeObject('House',
  {
    "latitude": 12.345,
    "longitude": 67.890,
    "name": "home"
  }
);

var door = new KnowledgeObject('Door',
  {
    "isOpen": false,
    "name": "Front door"
  }
);
```

4. Save the knowledge objects to the world model in the data store.

```
// Save the objects to the world model
Promise.all(
  [
    person.create(),
    house.create(),
    door.create()
  ]
).then(
  function (results) {
    console.log('All objects created\n\n');
```

5. Create relationships between the following objects in local memory:

   - The house and the front door.
   - The owner and the house. Use the `KnowledgeRelation` object. In the following code, in the `personToHouse` relationship, house `has-as-part` a front door. In the `houseToDoor` relationship, the person has `ownership` of the house.

```
// Create relations in local memory
var personToHouse = new KnowledgeRelation('ownership', person, house);
var houseToDoor = new KnowledgeRelation('has-as-part', house, door);
```

6. Save the relationship objects to the world model in the data store.

```
// Save relationships to the world model
Promise.all(
      [
        personToHouse.create(),
        houseToDoor.create()
      ]).then(
      function (results) {
        console.log('All relations created\n\n');
        runAgent();
      }
    );
  }
);
```

7. Create a proactive agent (`doorOpenAgent`) to react to the state change event.

```
// create the agents to connect to the Message Hub and subscribe to object update
events.
var doorOpenAgent = new Agent('object-update');

function runAgent() {
  Promise.all([
    doorOpenAgent.connect(),
  ]).then(function () {
    doorOpenAgent.subscribe();
    console.log('Subscription created\n\n');
  }, cleanup); //cleanup if the sub fails
}
```

8. Add a function to remove the objects and relations if creation of the world model does not complete successfully.

```
// Delete objects from the world model
function cleanup() {
  Promise.all(
    [
      person.delete(),
      house.delete(),
      door.delete()
    ]);
}
```

9. Add a function to update the status of the door to open when the function is called. The function checks that the door is closed before sending the update to the world model.

```
// Open the door
app.get('/openDoor', function (req, res) {
  KnowledgeObject.retrieve(door.id).then((doorObj) => {
    if (!doorObj.attributes.isOpen) {
      doorObj.attributes['isOpen'] = true;
      doorObj.update();
      res.status(200);
      res.send("opened door");
    } else {
      res.status(200);
      res.send("door was already open");
    }
  });
});
```

10. Add a function to update the status of the door to closed when the function is called. The function checks that the door is open before sending the update to the world model.

```
//Close the door
app.get('/closeDoor', function (req, res) {
  KnowledgeObject.retrieve(door.id).then((doorObj) => {
    if (doorObj.attributes.isOpen) {
      doorObj.attributes['isOpen'] = false;
      doorObj.update();
      res.status(200);
      res.send("closed door");
    } else {
      res.status(200);
      res.send("door was already closed");
    }
  });
});
```

11. Add a function to start the agent.

```
// Server Startup
const port = process.env.PORT || process.env.RULE_PORT || 8080;
app.listen(port, () => {
  console.log(`Agent REST service is alive!\nListening on port ${port}\n\n`)
});
module.exports.App = app;
```

12. Save your changes to the `homeSecurity.js` file.

> **What to do next?**
> Create the condition part of the rule.