BRAIN-IoT ROS Edge Node-Events

1. Introduction

This project is an implementation of ROS Edge Node based on the existing rososgi project. The ROS Edge Node provide the connectivity for ROS-based device within an OSGi environment, it exposes the ROS topics/services as OSGi services and integrates with the EventBus developed in BRAIN-IoT Fabric. Hence, the ROS Edge Node can be packaged as a smart behvior OSGi bundle, deployed in BRAIN-IoT Fabric as a BRAIN-IoT service, and communicate with other BRAIN-IoT services via EventBus. In this document, we will demonstrate in details about the events for the communication between the SmartBehavior and the RobotService in this project.

2. Basic Mechanism

The events inside this project are responsible for carrying commands and feedback to communicate between SmartBehavior and RobotService. And all the events will base on a father class like this:

```
public abstract class RobotCommand extends BrainIoTEvent {


       public static final int ALL_ROBOTS = -1;


       public int robotId;

}
```

And based on this father class, we have children classes like WriteGOTO, PickCart ... which will be specify in the next section. The instances of these children classes are the events that will be transmitted in the eventBus and consumed by the corresponding service.

To do the transmission, we can easily use the “deliver” method inside the eventBus class to send the event to the eventBus, like this:

```
@Reference

       private EventBus eventBus;

       ......

       eventBus.deliver(event);

       ......
```

To receive the events, we need to specify which events we want to consume before the service class using declarative service “@SmartBehaviourDefinition”. And then the corresponding events will be inject to the notify() method inside the service, where we can call the corresponding methods to execute the command. The receiver structure is like this:

```
@SmartBehaviourDefinition(

              consumed = {WriteGOTO.class,Cancel.class,
PickCart.class,PlaceCART.class,QueryState.class,CheckMarker.class },
            author = "LINKS", name = "Smart Robot",

            description = "Implements a remote Smart Robot.")

public class RobotService extends AbstractNodeMain implements
SmartBehaviour<RobotCommand>{

      ......

      @Override

      public void notify(RobotCommand event) {

            if (event instanceof WriteGOTO) {

                  WriteGOTO wgoto = (WriteGOTO) event;

                  worker.execute(() ->{

                        writeGOTO(wgoto.mission);

                        ......}

                  );

            }else if (event instanceof Cancel) {

             ......

}
```

3. Events

3.1 Information Events

This type of events is aim to get robot status information or do the job of information transmission. Basically this type of events are defined in couple, one is defined as a request event, the other is defined as a response event.

3.1.1 QueryState Event

Class:

```
public class QueryState extends RobotCommand {

        public int mission;

}
```

Description: Used to trigger a query request in RobotService and get a feedback of a QueryStateValueReturn event, containing the query result, to the SmartBehavior.

Usage: Specify the target robotId and mission after initialized the the class and then deliver the instance to the eventBus.

And the possible values and meanings are as below:

Value	mission
3	Query the GoTo action
7	Query the Pick action
11	Query the Place action

3.1.2 QueryStateValueReturn Event

Class:

```
public class QueryStateValueReturn extends RobotCommand {

        public int mission;

        public int value;

}
```

Description: Used to return the query result and the delayed action(see section 3.2) result to SmartBehavior.

Usage: Specify the target robotId, mission and value after initialized the the class and then deliver the instance to the eventBus.

And the possible values and meanings are different when the result is for a query command or a delayed action. Here only describe the situation when returning the result of the query command, the latter will be specify in section 3.2.

Value	mission

3	Same as 3.1.1
7	
11	

Value	value
0	error
1	finished
2	queued
3	running
4	paused
5	unknow

Where the “value” means the possible state of the action.

3.1.3 CheckMarker

Class:

```
public class CheckMarker extends RobotCommand {

        public int robot;

}
```

Description: Used to trigger a checkMarker request in RobotService and return a feedback of CheckValueReturn event, containing the check result to the SmartBehavior.

Usage: Specify the target robotId after initialized the the class and then deliver the instance to the eventBus.

3.1.4 CheckValueReturn

Class:

```
public class CheckValueReturn extends RobotCommand {

        public int value;

}
```

Description: Return the number of the markers to the SmartBehavior.

Usage: Specify the robotId and the value of markers number after initialized the the class and then deliver the instance to the eventBus.

3.2 Action Events

This type of events is aim to command the robot to execute actions. For the actions that needs time to finish,we call them the delay actions. And it should return a QueryStateValueReturn event to notify the SmartBehavior after the action is finished. For those that will finish immediately, there will be no return events.

3.2.1WriteGOTO Event

Class:

```
public class WriteGOTO extends RobotCommand {

    public int mission;

}
```

Description: Used to trigger the RobotService to execute a GOTO command in order to move the robot to a specific position which specified by the value of “mission”. Since GOTO action is a delayed action, it will to return a feedback event after the action is finished.

Usage: In the SmartBehavior specify the target robotId after initialized the the class and then deliver the instance to the eventBus. When the RobotService receive the event, execute the command and return a QueryStateValueReturn event to SmartBehavior after the action is finished.

And the possible values and meanings are as below:

	mission	value
0	X	error
1	Place center	finished
2	Place left	queued
3	Place right	running
4	Storage	paused
5	Unload	unknown

Where the mission is the key to the pre-defined target position.

3.2.2 PickCart Event

Class:

```
public class PickCart extends RobotCommand {

    public int cart;

}
```

Description: Used to trigger the RobotService to execute a Pick command in order to command the robot to pickup the cart with certain cartId.Since Pick action is a delayed action, it will to return a feedback event after the action is finished.

Usage: Similar with the usage of 3.2.1WriteGOTO Event. And the possible cart value is the pre-defined cartId in the warehouse.

3.2.3 PlaceCart Event

Class:

```
public class PlaceCART extends RobotCommand {

        public int cart;

}
```

Description: Used to trigger the RobotService to execute a Place command in order to command the robot to place the cart with certain cartId. Since Place action is a delayed action, it will to return a feedback event after the action is finished.

Usage: Similar with the usage of 3.2.1WriteGOTO Event. And the possible cart value is the pre-defined cartId in the warehouse.

3.2.4 Cancel Event

Class:

```
public class Cancel extends RobotCommand {

        public int mission;

}
```

Description: Used to trigger the RobotService to execute a cancel command to stop the running action. And the state of the canceled action will be finished.

Usage: Specify the robotId and the mission after initialized the the class and then deliver the instance to the eventBus.

And the possible values and meanings are as below:

Value	mission
3	Cancel GoTo action
7	Cancel Pick action
11	Cancel Place action