

grammar org.xtext.example.mydsl.MyDsl with org.eclipse.xtext.common.Terminals

import "platform:/resource/dk.itu.mdd.policyengine/model/PolicyEngine.ecore"

import "<http://www.eclipse.org/emf/2002/Ecore>" as ecore

Model returns Model:

```
{Model}
name=EString
'{
    (stateDefinition+=State (stateDefinition+=State)*)?
    (timers+=Timer (timers+=Timer)*)?
    (policyDefinition+=Policy (policyDefinition+=Policy)*)?
    (schedules+=Schedule (schedules+=Schedule)*)?
    ('room-type' predefinedRooms+=Room)*
    (buildings+=Building (buildings+=Building)*)?
}';
```

ActuatorType returns ActuatorType:

```
LightSwitchActuator | HumidifierActuator | DoorActuator | WindowActuator |
RadiatorActuator | AudioAlarmActuator | ElectricalSwitchActuator |
WaterValveActuator | GasValveActuator;
```

SensorType returns SensorType:

```
MotionSensor | TemperatureSensor | RainSensor | TouchSensor | LightSensor |
SmokeSensor | CO2Sensor | InfraredLightSensor | HumiditySensor | DoorSensor |
WindowSensor;
```

Statement returns Statement:

```
State | Timer | If;
```

Expression returns Expression:

```
Conjunction({BinaryExpression.leftExpr=current}
    operator=('||')
    rightExpr=Conjunction)*;
```

Conjunction returns Expression:

```
Comparison({ BinaryExpression.leftExpr=current}
    operator=('&&')
    rightExpr=Comparison)* ;
```

Comparison returns Expression:

```
Primary({BinaryExpression.leftExpr=current}
    operator=('='|'!='|'<='|'<'|'>='|'>')
    rightExpr=(Primary | ValueExpression))* ;
```

RoomInstance returns Expression:

```
RoomExpression({BinaryMethod.leftExpr=current}
    operator=('.')
    rightExpr=ComponentInstance)* ;
```

ComponentInstance returns Expression:

```
ComponentExpression({BinaryMethod.leftExpr=current}
    operator=('=')
    rightExpr=ValueExpression)* ;
```

SetStateActuator returns Expression:

```

SetValue({BinaryMethod.leftExpr=current}
        operator=('=')
        rightExpr=ValueExpression )* ;

```

```

Primary returns Expression:
    UnaryExpression | '(' Expression ')' | TimeExpression | StateExpression |
    SensorExpression;

```

```

Condition returns Expression:
    '(' Expression ')'
;

```

```

SetValue returns Expression:
    ActuatorExpression | StateExpression
;

```

```

Then returns Expression:
    ResetExpression | SetStateActuator | RoomInstance;

```

```

UnaryExpression returns Expression:
    {UnaryExpression} operator=('!')expr= Primary;

```

```

ValueExpression returns Expression:
    IntValue | BoolValue;

```

```

Component returns Component:
    Sensor | Actuator;

```

```

EString returns ecore::EString:
    STRING | ID;

```

```

Building returns Building:
    {Building}
    'building'
    name=EString
    '{'
        (accessControl=AccessControl)?
        (timers+=Timer (timers+=Timer)*)?
        (floors+=Floor (floors+=Floor)*)?
    '}' ;

```

```

Room returns Room:
    {Room}
    name=EString
    ('is-of-type' extends+=[Room] (',' extends+=[Room])*)?
    ('is-governed-by' policies+=[Policy] (',' policies+=[Policy])*
        ('during' during+=[Schedule] (',' during+=[Schedule])* | 'during-always'))?
    '{'
        (declareSensor+=Sensor (declareSensor+=Sensor)*)?
        (declareActuator+=Actuator (declareActuator+=Actuator)*)?
        (timers+=Timer (timers+=Timer)*)?
    '}' )?
;

```

```

Policy returns Policy:

```

```

{Policy}
'policy'
name=EString
('uses sensors' (usesSensors+=SensorType (',' usesSensors+=SensorType)*))?)?
('uses actuators' (usesActuators+=ActuatorType (',' usesActuators
+=ActuatorType)*))?)?
('uses rooms' usesRooms+=[Room] (',' usesRooms+=[Room])* )?
'is-implemented-by' '{' specifiedBy+=Statement (specifiedBy+=Statement)* '}'
;

```

```

State returns State:
{State}
'state' name=EString (valueState?= '=' EBoolean)?
;

```

```

Timer returns Timer:
{Timer}
'timer' name=EString;

```

```

Schedule returns Schedule:
{Schedule}
'schedule'
name=EString
('days' weekdays+=Weekdays ("," weekdays+=Weekdays)*)?
('from' from=Time 'to' to=Time)?
;

```

```

AccessControl returns AccessControl:
{AccessControl}
'AccessControl'
'{'
    ('accessControlSensors' '{' accessControlSensors+=SensorType ( ","
    accessControlSensors+=SensorType)* '}' )?
    ('accessControlDoorLockActuator' '{' accessControlDoorLockActuator
+=DoorActuator ( "," accessControlDoorLockActuator+=DoorActuator)* '}' )?
'}';

```

```

Floor returns Floor:
{Floor}
'floor'
name=EString
'{'
    ('room' rooms+=Room ( 'room' rooms+=Room)*)?
'}';

```

```

TemperatureSensor returns TemperatureSensor:
{TemperatureSensor}
'TemperatureSensor'
;

```

```

Actuator returns Actuator:
{Actuator}
'actuator' name=EString
'is a' actuatorTypes+=ActuatorType ( "," actuatorTypes+=ActuatorType)*
;

```

```

LightSwitchActuator returns LightSwitchActuator:

```

```
{LightSwitchActuator}  
'LightSwitchActuator'  
;
```

```
WindowActuator returns WindowActuator:  
    {WindowActuator}  
    'WindowActuator'  
;
```

```
HumidifierActuator returns HumidifierActuator:  
    {HumidifierActuator}  
    'HumidifierActuator'  
;
```

```
DoorActuator returns DoorActuator:  
    {DoorActuator}  
    'DoorActuator'  
;
```

```
RadiatorActuator returns RadiatorActuator:  
    {RadiatorActuator}  
    'RadiatorActuator'  
;
```

```
AudioAlarmActuator returns AudioAlarmActuator:  
    {AudioAlarmActuator}  
    'AudioAlarmActuator'  
;
```

```
ElectricalSwitchActuator returns ElectricalSwitchActuator:  
    {ElectricalSwitchActuator}  
    'ElectricalSwitchActuator'  
;
```

```
WaterValveActuator returns WaterValveActuator:  
    {WaterValveActuator}  
    'WaterValveActuator'  
;
```

```
GasValveActuator returns GasValveActuator:  
    {GasValveActuator}  
    'GasValveActuator'  
;
```

```
MotionSensor returns MotionSensor:  
    {MotionSensor}  
    'MotionSensor'  
;
```

```
DoorSensor returns DoorSensor:  
    {DoorSensor}  
    'DoorSensor'  
;
```

```
WindowSensor returns WindowSensor:  
    {WindowSensor}  
    'WindowSensor'  
;
```

```
RainSensor returns RainSensor:
    {RainSensor}
    'RainSensor'
;
```

```
TouchSensor returns TouchSensor:
    {TouchSensor}
    'TouchSensor'
;
```

```
LightSensor returns LightSensor:
    {LightSensor}
    'LightSensor'
;
```

```
SmokeSensor returns SmokeSensor:
    {SmokeSensor}
    'SmokeSensor'
;
```

```
CO2Sensor returns CO2Sensor:
    {CO2Sensor}
    'CO2Sensor'
;
```

```
InfraredLightSensor returns InfraredLightSensor:
    {InfraredLightSensor}
    'InfraredLightSensor'
;
```

```
HumiditySensor returns HumiditySensor:
    {HumiditySensor}
    'HumiditySensor'
;
```

```
Sensor returns Sensor:
    {Sensor}
    'sensor'
    name=EString
    ('is a' sensorTypes+=SensorType ( "," sensorTypes+=SensorType)*)?
;
```

```
If returns If:
    'if' cond=Condition
    '{'
        (then+=Then (then+=Then)*)?
        (elseif+=Statement (elseif+=Statement)*)?
    '}' ('else' '{'
        (then+=Then (then+=Then)*)?
        (else+=Statement (else+=Statement)*)? '}'
    )?;
```

```
TimeExpression returns TimeExpression:
    time=[Timer] 'reaches' timeAmount=EInt ('seconds' | 'minutes' | 'hours' |
    'days' );
```

```
ResetExpression returns ResetExpression:
```

```
'reset' reset=[Timer];
```

```
ComponentExpression returns ComponentExpression:  
    {ComponentExpression}  
    (instance=[Component])?;
```

```
StateExpression returns StateExpression:  
    {StateExpression}  
    'state-instance' '.'  
    (defineState=[State])?  
    ;
```

```
RoomExpression returns RoomExpression:  
    {RoomExpression}  
    'room-instance' '.'  
    (roomInstance=[Room])?  
    ;
```

```
SensorExpression returns SensorExpression:  
    {SensorExpression}  
    (sen=SensorType '.' 'value')?  
    ;
```

```
ActuatorExpression returns ActuatorExpression:  
    {ActuatorExpression}  
    (act=ActuatorType)?  
    '.' 'setValue'  
    ;
```

```
BoolValue returns BoolValue:  
    {BoolValue}  
    EBoolean  
    ;
```

```
IntValue returns IntValue:  
    {IntValue}  
    EInt  
    ;
```

```
EInt returns ecore::EInt:  
    '-'? INT;
```

```
EBoolean returns ecore::EBoolean:  
    'true' | 'false';
```

```
enum Weekdays:  
    MONDAY = 'Monday' | TUESDAY = 'Tuesday' | WEDNESDAY = 'Wednesday' | THURSDAY =  
    'Thursday' | FRIDAY = 'Friday' | SATURDAY = 'Saturday' | SUNDAY = 'Sunday'  
    ;
```

```
Time returns Time:  
    {Time}  
    (hours=EShort ':' minutes=EShort)?  
    ;
```

```
EShort returns ecore::EShort:  
    '-'? INT;
```