OWLS-FDplan: User Guide

a complete example

Elena Jaramillo, Anthony Heggen, Guangyi Chen

June 11, 2021

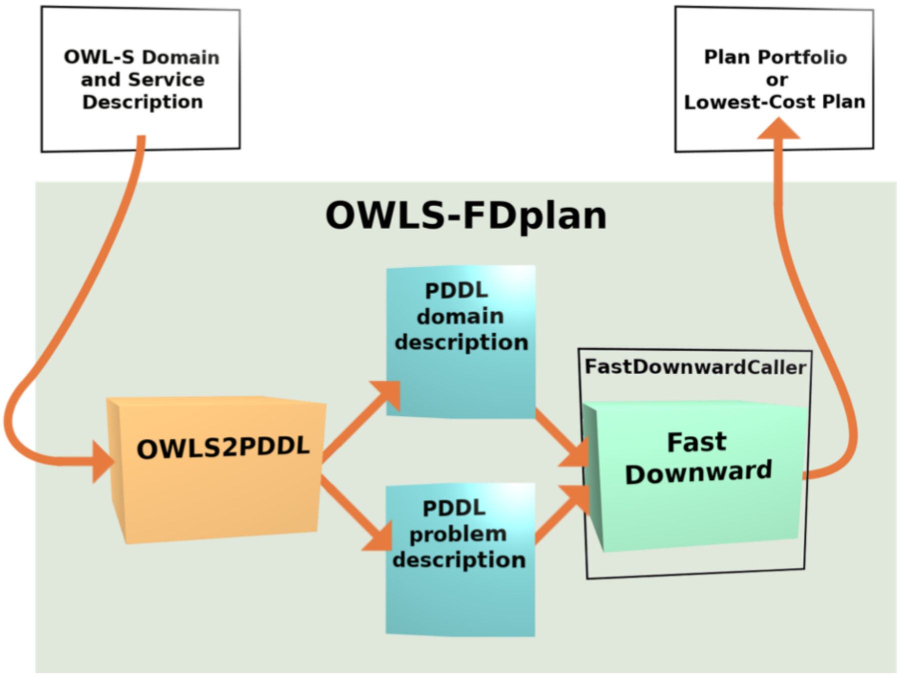


Figure 1: OWLS-FDplan Architecture

OWLS-FDplan is a service composition planner: Given a service composition problem expressed in OWL-S 1.1 as input, OWLS-FDplan produces a set of plans solving that problem. This document gives users a complete example to use the OWLS-FDplan. It consists of two steps: 1) using `OWLS2PDDL` to convert OWL files into PDDL files; 2) invoking `FastDownwardCaller` on this PDDL representation to produce the set of plans (OWL-S). The details are given in separate sections below.

1 OWL2SPDDL

OWLS2PDDL is a Java 1.8 tool that converts a problem and/or domain described in OWL to equivalent PDDL. In order to do so, OWLS2PDDL requires at least one of the following two sets of inputs:

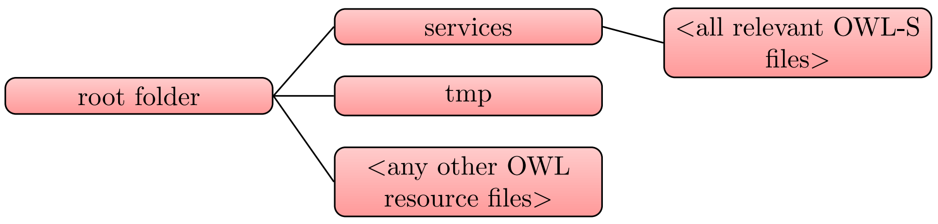
- An initial-state and goal-state OWL files
- OWL-S file(s)

If an initial-state and goal-state OWL are provided, then it is possible to output a PDDL problem; and if OWL-S files are provided, it is possible to output a PDDL domain. There is no need to provide both sets of inputs: one can produce PDDL problems and domains independent of each other.

The current version of OWLS2PDDL is able convert composed-service OWL-S using PDDL expressions to PDDL domains. Although OWLS2PDDL has gone through several iterations, its core functionalities remain unchanged and derived from the OWL API.

1.1 Local File System

The current version of OWLS2PDDL requires read/write access to a local directory. This directory must be hosted at http://127.0.0.1:8080 (see section 1.2), and should have the following structure:



The initial-state and goal-state OWL files are suggested to be placed directly under the root folder (<any other OWL resource files> in the structure). If the tmp directory does not exist, OWLS2PDDL will attempt to generate it. However, Windows 10 is known to block OWLS2PDDL's ability to write directories – in this case, it is easiest to manually create the tmp directory.

In this example, we are creating a root folder called Scallops, the structure is shown as Figure 2.

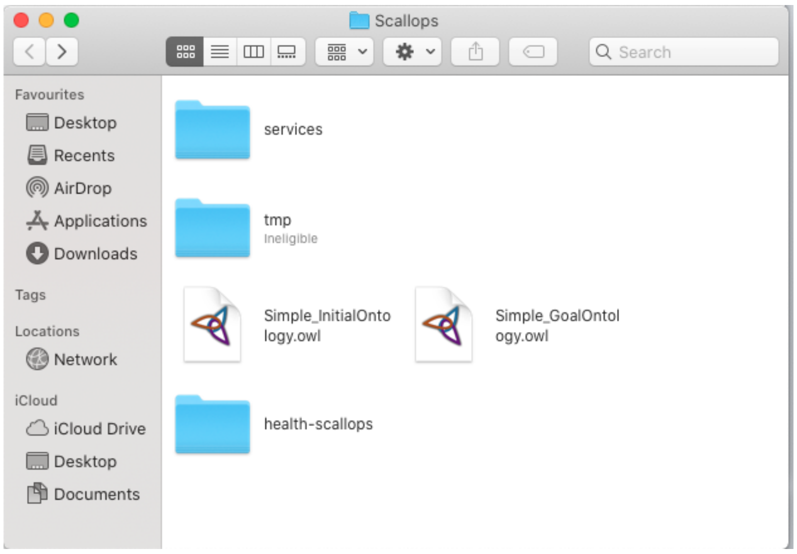
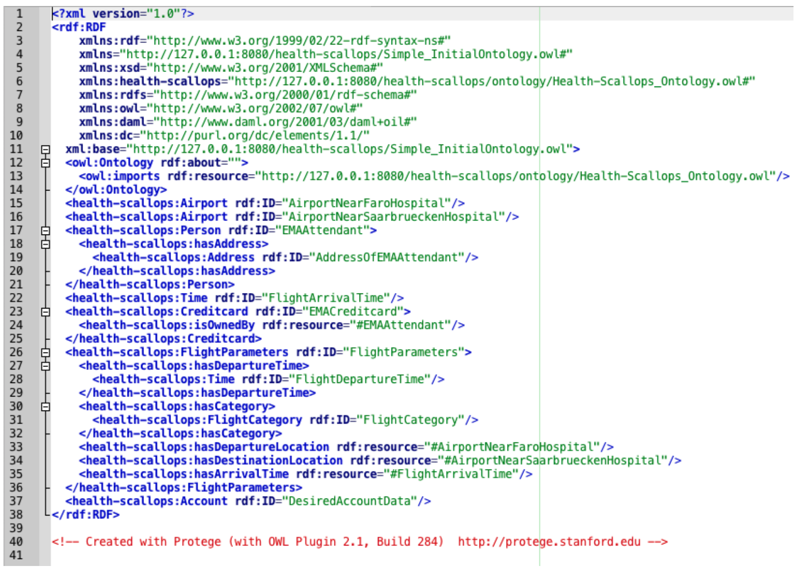
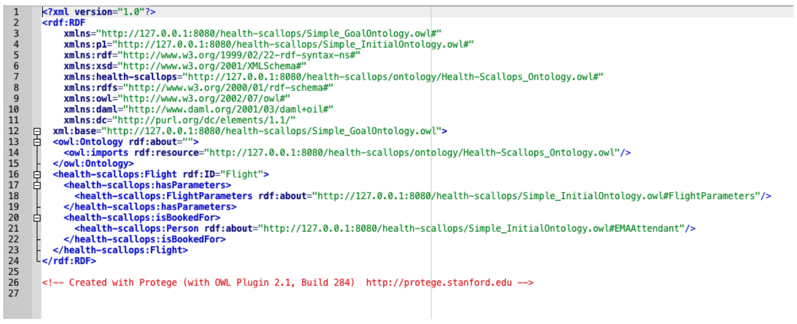


Figure 2: The example folder



```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns="http://127.0.0.1:8080/health-scallops/Simple_InitialOntology.owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:health-scallops="http://127.0.0.1:8080/health-scallops/ontology/Health-Scallops_Ontology.owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
  xml:base="http://127.0.0.1:8080/health-scallops/Simple_InitialOntology.owl">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://127.0.0.1:8080/health-scallops/ontology/Health-Scallops_Ontology.owl"/>
  </owl:Ontology>
  <health-scallops:Airport rdf:ID="AirportNearFaroHospital"/>
  <health-scallops:Airport rdf:ID="AirportNearSaarbrueckenHospital"/>
  <health-scallops:Person rdf:ID="EMAAttendant">
    <health-scallops:hasAddress>
      <health-scallops:Address rdf:ID="AddressOfEMAAttendant"/>
    </health-scallops:hasAddress>
  </health-scallops:Person>
  <health-scallops:Time rdf:ID="FlightArrivalTime"/>
  <health-scallops:Creditcard rdf:ID="EMACreditcard">
    <health-scallops:isOwnedBy rdf:resource="#EMAAttendant"/>
  </health-scallops:Creditcard>
  <health-scallops:FlightParameters rdf:ID="FlightParameters">
    <health-scallops:hasDepartureTime>
      <health-scallops:Time rdf:ID="FlightDepartureTime"/>
    </health-scallops:hasDepartureTime>
    <health-scallops:hasCategory>
      <health-scallops:FlightCategory rdf:ID="FlightCategory"/>
    </health-scallops:hasCategory>
    <health-scallops:hasDepartureLocation rdf:resource="#AirportNearFaroHospital"/>
    <health-scallops:hasDestinationLocation rdf:resource="#AirportNearSaarbrueckenHospital"/>
    <health-scallops:hasArrivalTime rdf:resource="#FlightArrivalTime"/>
  </health-scallops:FlightParameters>
  <health-scallops:Account rdf:ID="DesiredAccountData"/>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 2.1, Build 284)  http://protege.stanford.edu -->

```

Figure 3: Example initial-state owl file (`Simple_InitialOntology.owl`)



```
<?xml version="1.0"?>
<rdf:RDF
    xmlns="http://127.0.0.1:8080/health-scallops/Simple_GoalOntology.owl#"
    xmlns:p1="http://127.0.0.1:8080/health-scallops/Simple_InitialOntology.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:health-scallops="http://127.0.0.1:8080/health-scallops/ontology/Health-Scallops_Ontology.owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
  xml:base="http://127.0.0.1:8080/health-scallops/Simple_GoalOntology.owl">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://127.0.0.1:8080/health-scallops/ontology/Health-Scallops_Ontology.owl"/>
  </owl:Ontology>
  <health-scallops:Flight rdf:ID="Flight">
    <health-scallops:hasParameters>
      <health-scallops:FlightParameters rdf:about="http://127.0.0.1:8080/health-scallops/Simple_InitialOntology.owl#FlightParameters"/>
    </health-scallops:hasParameters>
    <health-scallops:isBookedFor>
      <health-scallops:Person rdf:about="http://127.0.0.1:8080/health-scallops/Simple_InitialOntology.owl#EMAAttendant"/>
    </health-scallops:isBookedFor>
  </health-scallops:Flight>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 2.1, Build 284)  http://protege.stanford.edu -->
```

Figure 4: Example Goal-state owl file (`Simple_GoalOntology.owl`)

Note that the `health-scallops` folder is the referenced service, it is not mandatory for all scenarios and can be customized accordingly. The details of example initial-state and goal-state OWL files are shown as Figure 3 and Figure 4.

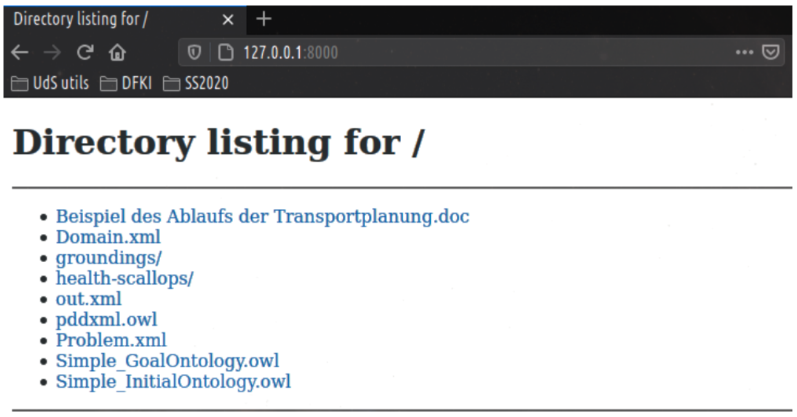
1.2 Setting Up a Local Server

OWLS2PDDL expects its services to be hosted on a webserver, and was developed with a python http server hosting these services. The CLI commands that follow will set up python 3's http server. Note that the root folder mentioned below is the one detailed in section 2.1.

Windows:
```
> cd path/to/root/folder
> python3 -m http.server 8000 --bind 127.0.0.1
```

Linux:
```
$ cd path/to/root/folder
$ python3 -m http.server 8000 --bind 127.0.0.1
```

If python 3 is not on your system's `PATH` variable, then it will be necessary to first add it before attempting the above commands. On both Windows 10 and Ubuntu, the ability to add python 3 to one's path is given during installation of python 3.



Directory listing for /

- Beispiel des Ablaufs der Transportplanung.doc
- Domain.xml
- groundings/
- health-scallops/
- out.xml
- pddxml.owl
- Problem.xml
- Simple_GoalOntology.owl
- Simple_InitialOntology.owl

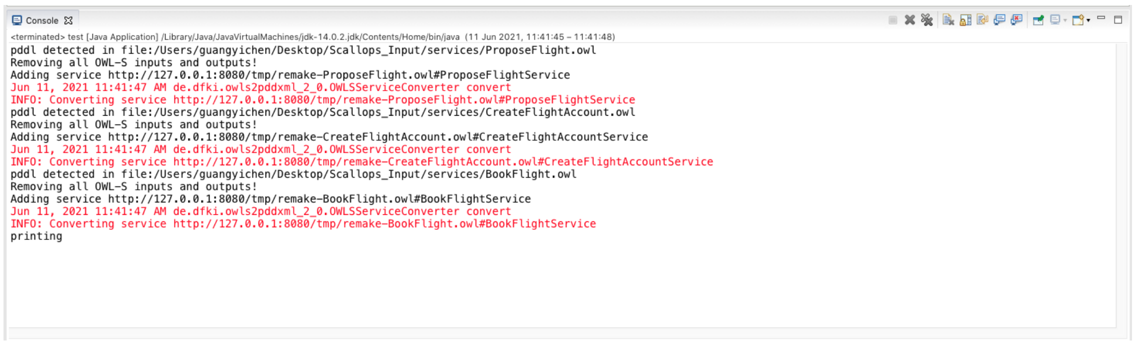
Figure 5: If hosting is successful, then visiting http://127.0.0.1:8000/ should bring up a directory resembling the above

1.3 Invoking OWLS2PDDL

Assuming that sections 1.1 and 1.2 have been completed, then adding the v1.3.2 `OWLS2PDDL` .jar to your build path, `OWLS2PDDL` can be invoked as follows:

```
import java.io.File;
import java.io.IOException;
import org.semanticweb.owl.model.OWLException;
import de.dfki.owls2pddxml_2_0.OWLS2PDDL;

   // set up the location to output pddl files
   public static final String PDDXML_DOMAIN_PRINT_LOCATION = "your-path/domain.xml";
   public static final String PDDXML_PROBLEM_PRINT_LOCATION = "your-path/problem.xml";
   public static final String PDDL_DOMAIN_PRINT_LOCATION = "your-path/domain.pddl";
   public static final String PDDL_PROBLEM_PRINT_LOCATION = "your-path/problem.pddl";


   OWLS2PDDL converter = new OWLS2PDDL("exampleDomainName",
        "your-path-to-root-folder/Scallops_Input");

   File initFile = new File("path-to-inital-ontology/Simple_InitialOntology.owl");
   File goalFile = new File("path-to-Goal-ontology/Simple_GoalOntology.owl");


   try {

   converter.addToInitialState(initFile.toURI());
   converter.addToGoalState(goalFile.toURI());

   File[] files = new
        File("/Users/guangyichen/Desktop/Scallops_Input/services").listFiles();

   for (File file : files)
   if (!file.isDirectory() && file.getName().contains(".owl")) {

      converter.addServices(file.toURI());

   }

   } catch(OWLException | IOException e) {

   e.printStackTrace();

   }


   // Get the files as strings. Pass "true" to make the files human-readable by remove
        URI syntax from actions, predicates, parameters, and effects
   String pddlDomain = converter.getDomain().makePDDLTextDocument(false,
        converter.getFilesWithPddl());
   String pddlProblem = converter.getProblem().makePDDLTextDocument(false);

   System.out.println("printing");

   FileWriter pddlDomainFileWriter;

   try {
      pddlDomainFileWriter = new FileWriter(PDDL_DOMAIN_PRINT_LOCATION);//new
          FileWriter(initFile.getAbsolutePath().split(initFile.getName())[0]
        +"_InitialOntology_Domain.xml");
     String pddlString = converter.getDomain().makePDDLTextDocument(false,
         converter.getFilesWithPddl());
     pddlDomainFileWriter.write(pddlString);
       pddlDomainFileWriter.close();
     } catch (IOException e) {

     e.printStackTrace();
   }

   FileWriter pddlProblemFileWriter;

   try {

   pddlProblemFileWriter = new FileWriter(PDDL_PROBLEM_PRINT_LOCATION);//new
       FileWriter(initFile.getAbsolutePath().split(initFile.getName())[0]+"_InitialOntology_PB.xml");
   pddlProblemFileWriter.write(converter.getProblem().makePDDLTextDocument(false));
   pddlProblemFileWriter.close();
   } catch (IOException e) {

     e.printStackTrace();
   }

   FileWriter domainFileWriter;

   try {
     domainFileWriter = new FileWriter(PDDXML_DOMAIN_PRINT_LOCATION);//new
         FileWriter(initFile.getAbsolutePath().split(initFile.getName())[0]
         +"_InitialOntology_Domain.xml");
     domainFileWriter.write(converter.getDomain().makeTextDocument());
     domainFileWriter.close();
   } catch (IOException e) {

      e.printStackTrace();
   }

   FileWriter problemFileWriter;

   try {

     problemFileWriter = new FileWriter(PDDXML_PROBLEM_PRINT_LOCATION);//new
         FileWriter(initFile.getAbsolutePath().split(initFile.getName())[0]+"_InitialOntology_PB.xml");
     problemFileWriter.write(converter.getProblem().makeTextDocument());
      problemFileWriter.close();

   } catch (IOException e) {

     e.printStackTrace();

   }
   }

}
```

If the run was successful, you may find the PDDL domain and problem files in your assigned directory. The console should have similar outputs as Figure 6. The example output PDDL files should be ready at your assigned location.



```
<terminated> test [Java Application] /Library/Java/JavaVirtualMachines/jdk-14.0.2.jdk/Contents/Home/bin/java  (11 Jun 2021, 11:41:45 – 11:41:48)
pddl detected in file:/Users/guangyichen/Desktop/Scallops_Input/services/ProposeFlight.owl
Removing all OWL-S inputs and outputs!
Adding service http://127.0.0.1:8080/tmp/remake-ProposeFlight.owl#ProposeFlightService
Jun 11, 2021 11:41:47 AM de.dfki.owls2pddxml_2_0.OWLSServiceConverter convert
INFO: Converting service http://127.0.0.1:8080/tmp/remake-ProposeFlight.owl#ProposeFlightService
pddl detected in file:/Users/guangyichen/Desktop/Scallops_Input/services/CreateFlightAccount.owl
Removing all OWL-S inputs and outputs!
Adding service http://127.0.0.1:8080/tmp/remake-CreateFlightAccount.owl#CreateFlightAccountService
Jun 11, 2021 11:41:47 AM de.dfki.owls2pddxml_2_0.OWLSServiceConverter convert
INFO: Converting service http://127.0.0.1:8080/tmp/remake-CreateFlightAccount.owl#CreateFlightAccountService
pddl detected in file:/Users/guangyichen/Desktop/Scallops_Input/services/BookFlight.owl
Removing all OWL-S inputs and outputs!
Adding service http://127.0.0.1:8080/tmp/remake-BookFlight.owl#BookFlightService
Jun 11, 2021 11:41:47 AM de.dfki.owls2pddxml_2_0.OWLSServiceConverter convert
INFO: Converting service http://127.0.0.1:8080/tmp/remake-BookFlight.owl#BookFlightService
printing
```

Figure 6: If `OWLS2PDDL` runs successfully, then console should have outputs resembling the above

2 FastDownwardCaller

`FastDownwardCaller` is a Java wrapper for Fast-Downward. `FastDownwardCaller` returns either a `String`, `Plan` (OWL-S), or `PlanProfile`; The `String` and `Plan` detail the shortest plan Fast-Downward can find from a given domain and problem pddl, while the `PlanProfile` includes all plans found by Fast Downward in the given time. Thus, three components are necessary to use `FastDownwardCaller`:

(a) `FastDownwardCaller.jar`

(b) A compiled version of Fast-Downward (see section 2.1 for installation)

(c) A problem and domain PDDL

2.1 Installing Fast-Downward

FastDownwardCaller uses Fast-Downward, hence it is necessary to pre-install Fast-Downward in your local environment. To achieve this first go to the Repository, clone or download the codes and unzip it to your assigned location. Then open your terminal, type the following commands:

```
$ cd path-to-dir/downward-main
$ ./build.py release
```

2.2 Setting Up a Local Server

In consistent with OWLS2PDDL, FastDownwardCaller expects its services to be hosted on a web-server. To do so firstly goes to the service directory in the terminal, then use python 3's server with the following command:

```
$ python3 -m http.server 8000 --bind 127.0.0.1
```

Note that this service directory needs to contain service OWL-S files. In this example, we are hosting on the same directory as in section 1.1.

2.3 Invoking FastDownwardCaller V1.2

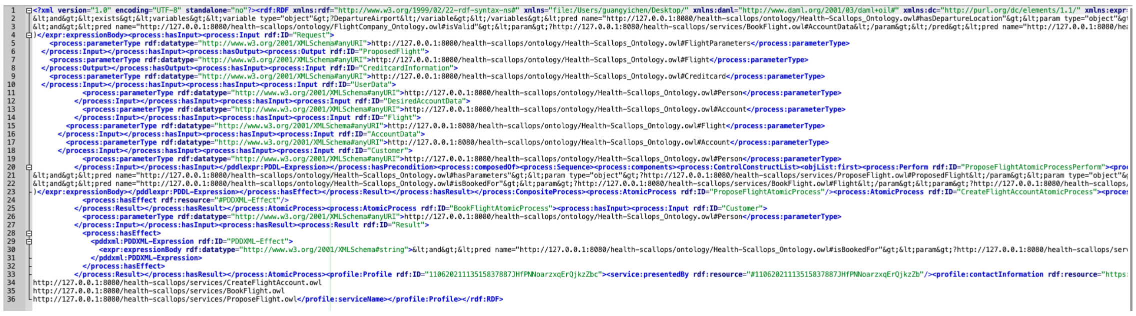
Firstly, make sure to include the jsoup-1.13.1.jar as Referenced Libraries and check if it is located in the correct build path.

Then, assuming that `FastDownwardCaller.jar` is on one's buildpath, it can be invoked as follows:

```java
import java.io.File;
import java.net.URI;

import org.jsoup.*;
import org.w3c.dom.Document;

import de.dfki.fastdownwardcaller.*;

   // initialize pointing to FastDownward directory
   FastDownwardCaller caller = new FastDownwardCaller("your-path/downward-main");

   try {
```

```
            // pass Strings representing domain and problem pddls
            // here you may use your outputs from OWLS2PDDL in the previous step for
                problem.pddl and domain.pddl files
            caller.readInProblem(new File("your-path-to-problem_pddl/problem.pddl"));
            caller.readInDomain(new File("your-path-to-domain_pddl/domain.pddl"));

        } catch (Exception e) {

            e.printStackTrace();

        }

        // Get a composed service as a w3c Document
        Plan lowestCostPlan = caller.getLowestCostPlan(true);

        File outputDirectory = new File("/Users/guangyichen/Desktop/");
        URI compositeServiceUri = outputDirectory.toURI();

        CompositeService actionToService = new CompositeService(lowestCostPlan,
            compositeServiceUri);
        Document composedService = actionToService.getDocument();

    }
}
```

If FastdownwardCaller runs successfully, the output plan file should be available in the appointed location. In this example, the file details are shown as Figure 7.



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?><rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns="file:/Users/guangyichen/Desktop/" xmlns:daml="http://www.daml.org/2001/03/daml+oil#" xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:expr
&lt;and&gt;&lt;exists&gt;&lt;variables&gt;&lt;variable type="object"&gt;?DepartureAirport&lt;/variable&gt;&lt;/variables&gt;&lt;pred name="http://127.0.0.1:8080/health-scallops/ontology/Health-Scallops_Ontology.owl#hasDepartureLocation"&gt;&lt;param type="object"&g
&lt;and&gt;&lt;pred name="http://127.0.0.1:8080/health-scallops/ontology/FlightCompany_Ontology.owl#isValid"&gt;&lt;param&gt;?http://127.0.0.1:8080/health-scallops/services/BookFlight.owl#AccountData&lt;/param&gt;&lt;/pred&gt;&lt;pred name="http://127.0.0.1:8080/hea
)</expr:expressionBody><process:hasInput><process:Input rdf:ID="Request">
  <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">http://127.0.0.1:8080/health-scallops/ontology/Health-Scallops_Ontology.owl#FlightParameters</process:parameterType>
 </process:Input></process:hasInput><process:hasOutput><process:Output rdf:ID="ProposedFlight">
  <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">http://127.0.0.1:8080/health-scallops/ontology/Health-Scallops_Ontology.owl#Flight</process:parameterType>
 </process:Output></process:hasOutput><process:hasInput><process:Input rdf:ID="CreditcardInformation">
  <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">http://127.0.0.1:8080/health-scallops/ontology/Health-Scallops_Ontology.owl#Creditcard</process:parameterType>
 </process:Input></process:hasInput><process:hasInput><process:Input rdf:ID="UserData">
      <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">http://127.0.0.1:8080/health-scallops/ontology/Health-Scallops_Ontology.owl#Person</process:parameterType>
     </process:Input></process:hasInput><process:hasInput><process:Input rdf:ID="DesiredAccountData">
      <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">http://127.0.0.1:8080/health-scallops/ontology/Health-Scallops_Ontology.owl#Account</process:parameterType>
     </process:Input></process:hasInput><process:hasInput><process:Input rdf:ID="Flight">
    <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">http://127.0.0.1:8080/health-scallops/ontology/Health-Scallops_Ontology.owl#Flight</process:parameterType>
   </process:Input></process:hasInput><process:hasInput><process:Input rdf:ID="AccountData">
    <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">http://127.0.0.1:8080/health-scallops/ontology/Health-Scallops_Ontology.owl#Account</process:parameterType>
   </process:Input></process:hasInput><process:hasInput><process:Input rdf:ID="Customer">
      <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">http://127.0.0.1:8080/health-scallops/ontology/Health-Scallops_Ontology.owl#Person</process:parameterType>
     </process:Input></process:hasInput></pddlexpr:PDDL-Expression></process:hasPrecondition><process:composedOf><process:Sequence><process:components><process:ControlConstructList><objList:first><process:Perform rdf:ID="ProposeFlightAtomicProcessPerform"><proc
&lt;and&gt;&lt;pred name="http://127.0.0.1:8080/health-scallops/ontology/Health-Scallops_Ontology.owl#hasParameters"&gt;&lt;param type="object"&gt;?http://127.0.0.1:8080/health-scallops/services/ProposeFlight.owl#ProposedFlight&lt;/param&gt;&lt;param type="object"&g
&lt;and&gt;&lt;pred name="http://127.0.0.1:8080/health-scallops/ontology/Health-Scallops_Ontology.owl#isBookedFor"&gt;&lt;param&gt;?http://127.0.0.1:8080/health-scallops/services/BookFlight.owl#Flight&lt;/param&gt;&lt;param&gt;?http://127.0.0.1:8080/health-scallops/
)</expr:expressionBody></pddlexpr:PDDL-Expression></process:hasEffect></process:Result></process:hasResult></process:CompositeProcess><process:AtomicProcess rdf:ID="ProposeFlightAtomicProcess"/><process:AtomicProcess rdf:ID="CreateFlightAccountAtomicProcess"><proces
      <process:hasEffect rdf:resource="#PDDXML-Effect"/>
     </process:Result></process:hasResult></process:AtomicProcess><process:AtomicProcess rdf:ID="BookFlightAtomicProcess"><process:hasInput><process:Input rdf:ID="Customer">
      <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">http://127.0.0.1:8080/health-scallops/ontology/Health-Scallops_Ontology.owl#Person</process:parameterType>
     </process:Input></process:hasInput><process:hasResult><process:Result rdf:ID="Result">
      <process:hasEffect>
       <pddxml:PDDXML-Expression rdf:ID="PDDXML-Effect">
        <expr:expressionBody rdf:datatype="http://www.w3.org/2001/XMLSchema#string">&lt;and&gt;&lt;pred name="http://127.0.0.1:8080/health-scallops/ontology/Health-Scallops_Ontology.owl#isBookedFor"&gt;&lt;param&gt;?http://127.0.0.1:8080/health-scallops/serv
       </pddxml:PDDXML-Expression>
      </process:hasEffect>
     </process:Result></process:hasResult></process:AtomicProcess><profile:Profile rdf:ID="11062021113515837887JHfPNNoarzxqErQjkzZbc"><service:presentedBy rdf:resource="#11062021113515837887JHfPNNoarzxqErQjkzZb"/><profile:contactInformation rdf:resource="https:
http://127.0.0.1:8080/health-scallops/services/CreateFlightAccount.owl
http://127.0.0.1:8080/health-scallops/services/BookFlight.owl
http://127.0.0.1:8080/health-scallops/services/ProposeFlight.owl</profile:serviceName></profile:Profile></rdf:RDF>
```

Figure 7: Example output plan of `FastDownwardCaller`

3 Remarks

When preparing the OWL-S services files into the `services` folder, the following points should be addressed:

Currently, the converter cannot handle “local variables” in the OWL-S services. It can only use the variables defined by the International standard.

Also, the converter requires that the `inputs/outputs/pre-condition/effects` be “described” inside either the profile or the process.