Spatial Extension for Operational Decision Manager

Requirements

ODM v8.5.1, v8.6 (with corresponding supported versions of eclipse for rule designer)

Java 1.6+

Installation

Extract the archive into a folder on your drive. The root folder of the support pack contains the following:

doc – contains documentation for the support pack, - the user guide and the java API

samples – sample projects

com_ibm_geospatial_bom.zip – the zipped project containing the binaries and other artifacts for the spatial extension

Using Spatial Extension

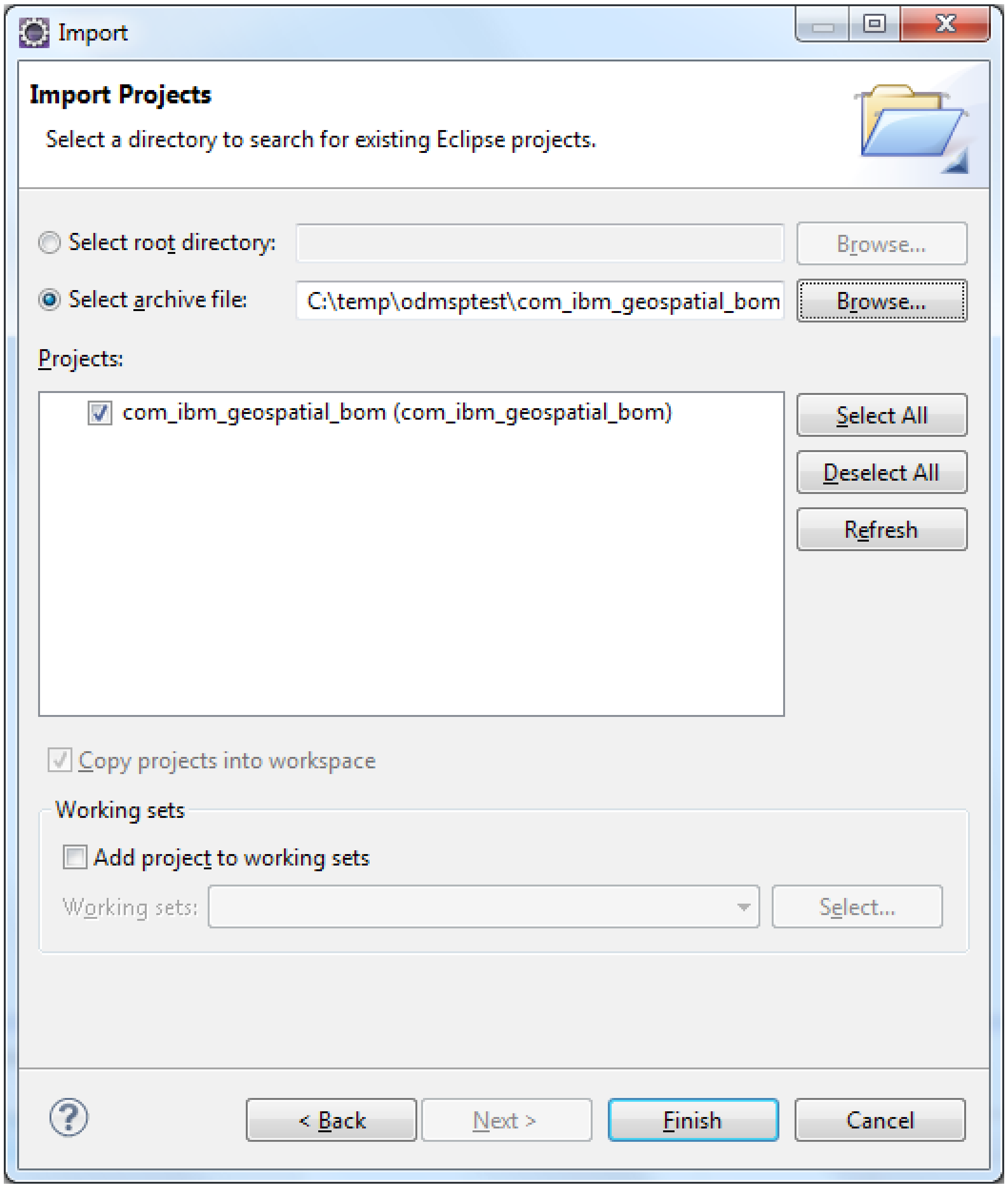
Using ODM spatial extension is simple: you should import our eclipse project into your eclipse workspace and refer to its BOM and java libraries from your rule or java projects. The following sections provide more detailed explanation of these steps.

Import spatial extension project

In your rule designer eclipse environment, import the spatial extension project using the following steps:

1. In the File menu, choose ‘Import…’
2. In the import wizard, select ‘Existing Projects into Workspace’ and press Next

3. On the ‘Import Projects’ wizard page, choose the ‘Select archive file’ and browse to the com_ibm_geospatial_bom.zip (see screenshot below):



Import

Import Projects

Select a directory to search for existing Eclipse projects.

Select root directory:

Select archive file: C:\temp\odmsptest\com_ibm_geospatial_bom

Projects:

[✓] com_ibm_geospatial_bom (com_ibm_geospatial_bom)

[✓] Copy projects into workspace

Working sets

[ ] Add project to working sets

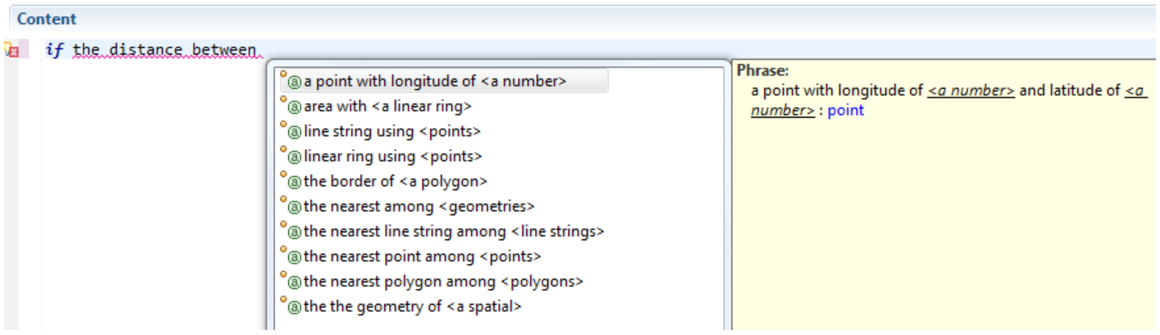
Working sets:

4. Make sure the com_ibm_geospatial_bom project is selected and press ‘Finish’

The geospatial bom project will be imported and should now appear in the project explorer (or rule explorer, depending on the current Eclipse perspective) view in your eclipse workbench. Now you’re all set for building your rule applications with geospatial rules.

Using geospatial BOM in rule projects

To use geospatial data types and operations in your rules, add a reference to the com_ibm_geospatial_bom project in your rule project. Once done that, you should see geospatial constructs available in your BOM editor or rule designer for your rule project (see screenshot).



Content

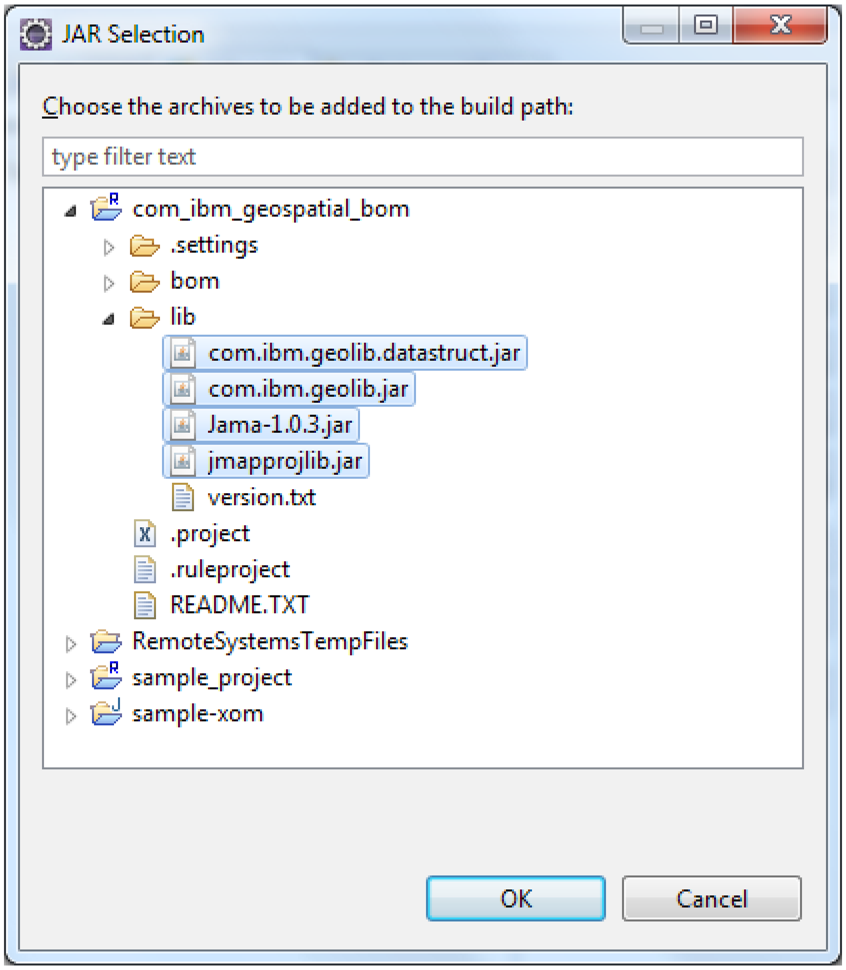
if the distance between

- a point with longitude of <a number>
- area with <a linear ring>
- line string using <points>
- linear ring using <points>
- the border of <a polygon>
- the nearest among <geometries>
- the nearest line string among <line strings>
- the nearest point among <points>
- the nearest polygon among <polygons>
- the the geometry of <a spatial>

Phrase:
a point with longitude of <a number> and latitude of <a number> : point

Using geospatial API in Java projects

You can use geospatial java API in your Java projects by adding the geospatial libraries to your project's class path. The libraries are located in the lib folder of the com_ibm_geospatial_bom project:



JAR Selection

Choose the archives to be added to the build path:

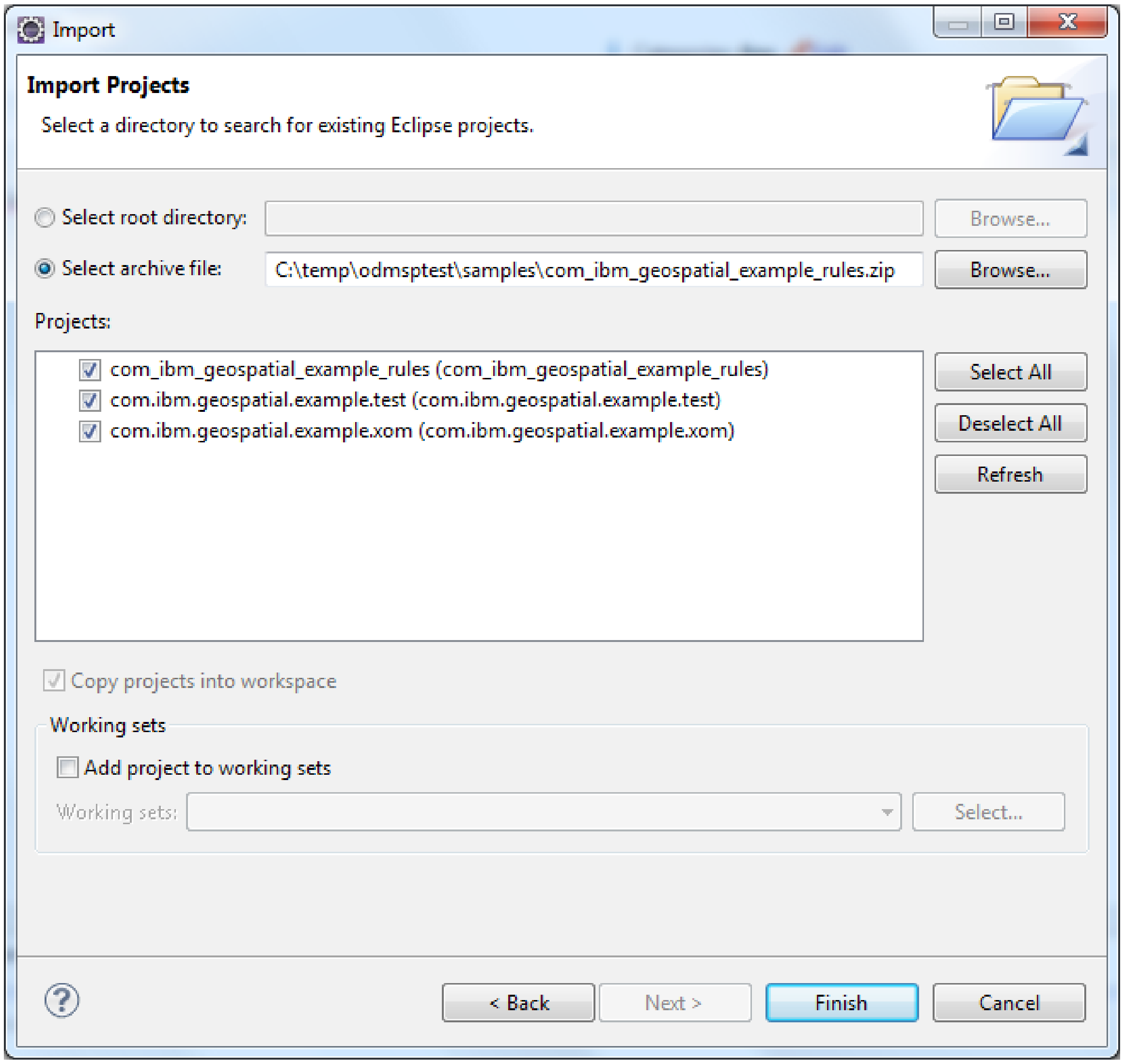
type filter text

```
com_ibm_geospatial_bom
    .settings
    bom
    lib
        com.ibm.geolib.datastruct.jar
        com.ibm.geolib.jar
        Jama-1.0.3.jar
        jmapprojlib.jar
        version.txt
    .project
    .ruleproject
    README.TXT
RemoteSystemsTempFiles
sample_project
sample-xom
```

Sample projects included in with the spatial extension package provide illustration and example for how to build a rule application with our geospatial libraries and BOM.

Sample projects

To work with sample projects simply import the projects contained in the samples/com_ibm_geospatial_example_rules.zip file:


Import

Import Projects

Select a directory to search for existing Eclipse projects.

( ) Select root directory:

(●) Select archive file: C:\temp\odmsptest\samples\com_ibm_geospatial_example_rules.zip

Projects:

- ☑ com_ibm_geospatial_example_rules (com_ibm_geospatial_example_rules)
- ☑ com.ibm.geospatial.example.test (com.ibm.geospatial.example.test)
- ☑ com.ibm.geospatial.example.xom (com.ibm.geospatial.example.xom)

☑ Copy projects into workspace

Working sets

☐ Add project to working sets

Working sets:

There are three sample projects, as follows:

- com_ibm_geospatial_example_rules – a rule project containing samples of geospatial rules
- com.ibm.geospatial.example.xom – the XOM for the rule project
- com.ibm.geospatial.example.test – a unit test (junit) project containing code to run and test the rules

To verify the setup we've done so far run the junit tests in the com.ibm.geospatial.example.test project. The tests use ruleset archives produced from the rules in the rules project. The test project you imported in the previous step contains the rule set jar archives. You can override them with new archives produced in your rule designer environment. To do that export the rules in the com_ibm_geospatial_example_rules project producing three rule set archives under com.ibm.geospatial.example.test project folder named testrules1.jar, testrules2.jar, and testrules3.jar using the com_ibm_geospatial_example_rules/queries/testN_rules.qry queries as rule extractors respectively.

In the following we'll walk you through the sample rules explaining the use of geospatial primitives.

Data model and rule set parameters for sample rules

The BOM for the sample rule project defines the following types:

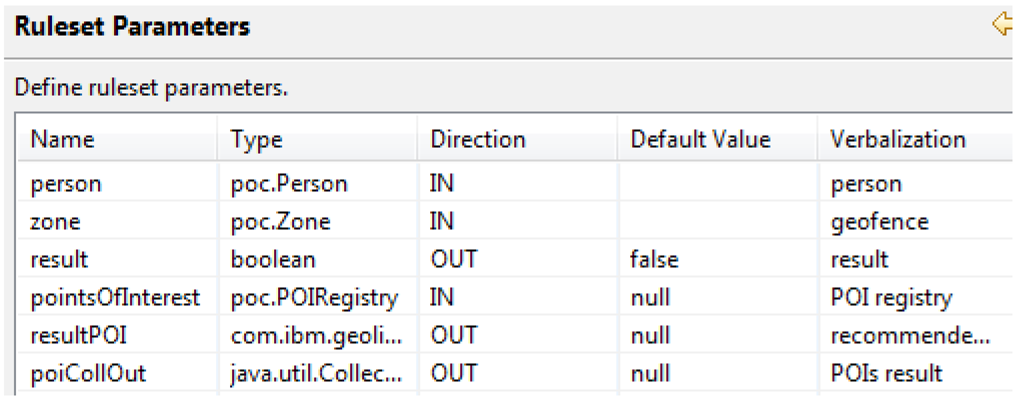
Person – represents a person. Besides such attributes as name and age a person also has a 'location' that holds the current geographic position of that person. The location is represented by a geographic point (com.ibm.geolib.geom.Point). The Point type is defined in the spatial extension BOM along with other data types representing geometries used to model geographic shape of a spatial object.

POI – a Point of Interest. POI class in the sample BOM extends com.ibm.geolib.geom.Spatial – a base interface defined in geospatial BOM to represent spatial objects – i.e. objects that have a geometry. The Spatial interface declares a single method – getGeometry() used to obtain the geometry of the object.

POIRegistry – a collection of POIs.

Zone – a named area. The 'area' attribute of a zone is of type com.ibm.geolib.geom.Polygon.

The rule set parameters for our sample rule project are as follows:


Ruleset Parameters

Define ruleset parameters.

Name	Type	Direction	Default Value	Verbalization
person	poc.Person	IN		person
zone	poc.Zone	IN		geofence
result	boolean	OUT	false	result
pointsOfInterest	poc.POIRegistry	IN	null	POI registry
resultPOI	com.ibm.geoli...	OUT	null	recommende...
poiCollOut	java.util.Collec...	OUT	null	POIs result

Rule: test1_person_inside_zone

This rule checks if the person is located inside the geofence (a zone), given that the location of the person as well as the geofence area are provided as the rule set input parameters. The geospatial primitive used in this rule is ‘contains’ which a method of a geometry and has the following syntax:

{this geometry} contains {another geometry}

```
if
    the area of geofence contains the location of person
then

    …
```

Rule: test2_nearest_grocery_store

The rule finds the nearest (to the person) POI of category ‘grocery store’ among the POIs passed in the POI Registry and returns this POI if its distance from the person is no more than 5 km:

```
definitions
    set 'all of category' to the points of interests in 'POI registry'
          where the category of each POI is "Grocery Store" ;

    set 'nearest store' to a POI in 'all of category' where this POI is the
nearest object to person among 'all of category' ;
if
    'nearest store' is not null
    and the distance between the the geometry of 'nearest store' and the
location of person in kilometers is at most 5
then
    set 'recommended POI' to 'nearest store' ;
    print "Found a store: " + the name of 'nearest store' ;
else
    set result to false ;
    print "store was not found" ;
```

In this example two geospatial primitives were used – the ‘nearest’ and the ‘distance’. The ‘nearest’ method returns a Spatial object among the collection of spatial objects passed as input. The distance method computes the distance between two geometries represented in the specified units of length (e.g. miles, kilometers, etc.).

Rule: test3_person_on_border

Sometimes it is required to detect either a spatial object is on a border of some area, or on a path. Because computing that a geometry is exactly (mathematically) on the border is impossible (and not required in the real life scenarios) we operate with borders of a non-zero width, and use the ‘distance’ operator to compute the result:

```
if
    the distance between the location of person and the border of the area of
geofence in meters is less than 10
then
    set result to true ;
    print the name of person + " is on the border of " + the name of
geofence ;
else
      set result to false ;
    print the name of person + " is NOT on the border of " + the name of
geofence ;
```

Rule: test4_all_pois_nearby

This short rule finds all POIs within the given distance from the person:

```
then set 'POIs result' to all objects from the points of interests in 'POI
registry' within distance of 5 kilometers from person ;
```

Using Java API

The java library implementing the XOM for geospatial BOM is also available for java developers. The documentation of Java API is included in form of javadoc HTML files.