**PDP User Guide**

**(draft)**

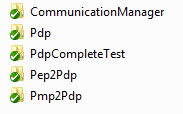
|  |  |  |
| --- | --- | --- |
| Revision history | | |
| *Date* | *Author* | *Comment* |
| 2013-07-10 | Stoimenov | Draft version |
|  |  |  |

**Notice:**

This guide is intended for developers who want to use Pep2Pdp and Pmp2Pdp stubs to communicate with PDP. If you want to further develop PDP module check “PDP Developers Guide”.

# Building PDP and stubs

PDP, PEP\_TO\_PDP stub and PMP\_TO\_PDP stub are coded in JAVA. Maven is used for project management. There is a parent project called PDP and there are several child projects. The project structure is flat which means that both parent and child projects are at the same level in the file system hierarchy. This is convenient when the projects are imported in eclipse. The following picture shows all projects:



To build the necessary stubs, open terminal, go to Pdp, and execute the following command:

mvn clean package

Notice: Make sure you have maven installed on your machine.

The jars representing the stubs are placed in the folders *Pep2Pdp/target* and *Pmp2Pdp/target* and are named *pep2pdp-1.0.jar* and *pmp2pdp-1.0.jar*.

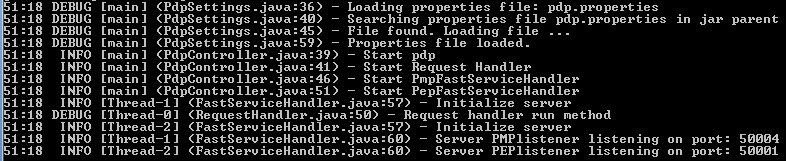
# Configuring and starting PDP

Follow the steps in section “Building PDP and stubs” to compile and package the code. Copy the file CommunicationManager/target/ CommunicationManager-1.0-jar-with-dependencies.jar to some empty folder. Copy files CommunicationManager/pdp.properties and CommunicationManager/src/main/resources/log4j.properties to the same folder.

Start the PDP server by executing the following command in the terminal:

# *java -Dlog4j.configuration=file:./log4j.properties -jar CommunicationManager-1.0-jar-with-dependencies.jar*

After executing this command you should see output in the terminal similar to this:



The PDP is now ready to accept connections from PEP and PMP modules.

# Using PEP\_TO\_PDP stubs

If you develop a PEP module, in order to establish a connection to the PDP and to communicate with it, you will need *pep2pdp-1.0.jar* and *log4j-1.2.17.jar* in the build/class path.

A practical example of the communication between PEP and PDP is contained in the project *Pep2Pdp* in the jUnit test *TestPep2PdpCommunication.java*.

# Using PMP\_TO\_PDP stubs

Similarly, if you develop a PMP module, you can establish a connection to the PDP and to communicate with it by using code from *pmp2pdp-1.0.jar* which should be included the build/class path. Additionally, *log4j-1.2.17.jar* must be included in the build/class path.

A practical example of the communication between PMP and PDP is contained in the project *Pmp2Pdp* in the jUnit test *TestPmp2PdpCommunication.java*.

# Communicating with PDP in a programming language other than JAVA

When developing PMP and PEP modules in Java, developers can use provided stubs to communicate with PDP.

If the programming language of PMP and/or PEP is not JAVA, then the following protocol must be implemented:

When writing to the socket output stream:

1. First byte denotes the method that will be called on the PDP
2. Following are the method parameters. Google Protocol Buffer provides a method called:  
   writeDelimitedTo(OutputStream outputStream);  
    on the GPB objects. This method can be used to write the message data to an output stream. The method will first write the size of the message and then the message bytes.
3. On the other side of the communication, received bytes that should be interpreted as corresponding Google Protocol Buffer object. For that purpose the following method can be used:  
   *GoogleProtocolBufferConcreteClass gpObject = GoogleProtocolBufferConcreteClass.parseDelimitedFrom(InputStream inputStream);*  
     
    to parse the object from the input stream.

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface name** | **Method name** | **Data to write to the output stream** | **Data to receive** |
| IPep2PdpFast | IResponse notifyEvent(IEvent event) | 1. One byte (any value), currently not used 2. Instance of GpEvent | Instance of GpResponse |
| IPmp2PdpFast | EStatus deployMechanism(IMechanism mechanism) | 1. One byte, value 1 2. Instance of GpMechanism | Instance of GpStatus |
| IMechanism exportMechanism(String par) | 1. One byte, value 2 2. Instance of GpString | Instance of GpMechanism |
| EStatus revokeMechanism(String par) | 1. One byte, value 3 2. Instance of GpString | Instance of GpStatus |