This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 640351.
## Change Record

<table>
<thead>
<tr>
<th>Issue</th>
<th>Date</th>
<th>Section, Page</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>31.08.2015</td>
<td></td>
<td>Initial draft issue</td>
</tr>
<tr>
<td>1.1</td>
<td>01.10.2015</td>
<td></td>
<td>First stable version that includes all operations description and the full installation procedure. Also implemented comments provided by Airbus dated 27th July 2015.</td>
</tr>
<tr>
<td>1.2</td>
<td>26.10.2015</td>
<td>All</td>
<td>Implementation of comments into various sections, which are raised by consortium and during a dedicated telecon.</td>
</tr>
<tr>
<td>1.3 interim2</td>
<td>21.01.2016</td>
<td></td>
<td>Updated with the latest implementations for the final delivery.</td>
</tr>
<tr>
<td>1.4</td>
<td>06.07.2017</td>
<td></td>
<td>Final edition with minor changes.</td>
</tr>
</tbody>
</table>
Table of Contents

1 Introduction...........................................................................................................................................5
  1.1 Scope...............................................................................................................................................5
  1.2 List of Abbreviations.........................................................................................................................5
  1.3 Applicable Documents .....................................................................................................................6
  1.4 Reference Documents ......................................................................................................................6
  1.5 Standards.........................................................................................................................................6

2 System Overview ...................................................................................................................................7
  2.1 Project background............................................................................................................................7
  2.2 NEO Physical Properties Database and Dynamical Web Interface ................................................8
    2.2.1 The ESA NEO Coordination Centre Physical Properties database .................................. 9

3 Overall Architecture ............................................................................................................................12
  3.1 Data Model......................................................................................................................................14
    3.1.1 OpenData Repository.................................................................................................................14
    3.1.2 NEO Physical Properties Database..........................................................................................16
  3.2 SOAP Middleware Interface ...........................................................................................................18
    3.2.1 JAX-WS SOAP API usage.......................................................................................................18
    3.2.2 Data Access Object usage........................................................................................................18
  3.3 Dynamical Web Interface ................................................................................................................19
  3.4 Architectural Open Points .................................................................................................................19

4 Implementation Technologies ...............................................................................................................20

Appendix A: Installation procedure ..........................................................................................................21
  4.1 Installation Pre-requisites ...............................................................................................................21
    4.1.1 Operating System .....................................................................................................................21
    4.1.2 Java VM....................................................................................................................................21
    4.1.3 Apache Maven..........................................................................................................................21
    4.1.4 MySQL Relational Database.....................................................................................................22
    4.1.5 Apache Tomcat..........................................................................................................................23
  4.2 Build procedure from sources .........................................................................................................24
    4.2.1 Dynamical Web Interface .........................................................................................................24
  4.3 Installation from binaries ...............................................................................................................32
    4.3.1 NEO Physical Properties Database ..........................................................................................32
    4.3.2 Dynamical Web Interface .........................................................................................................33
  4.4 Confidence Test ...............................................................................................................................34
  4.5 Maintenance Tasks ...........................................................................................................................36
    4.5.1 Logs ..........................................................................................................................................36
    4.5.2 Controlling the execution of the daemon..................................................................................36
1 Introduction

1.1 Scope

The scope of this document is to provide a complete view on the usage of the NEOShield-2 NEO Physical Properties Database, describing how to install, configure and operate the system. The manual is structured as follows:

- Section 2 provides an overview of the project scope.
- Section 3 describes the top-level architecture and the set of components of the system.
- Section 4 lists the third-party technologies used in the implementation of the system.
- Appendix A details the steps to install, deploy and test the system.

1.2 List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD</td>
<td>Applicable Document</td>
</tr>
<tr>
<td>API</td>
<td>Application Program Interface</td>
</tr>
<tr>
<td>DAO</td>
<td>Data Access Object</td>
</tr>
<tr>
<td>DMZ</td>
<td>Demilitarised Zone</td>
</tr>
<tr>
<td>EARN</td>
<td>European Asteroid Research Node</td>
</tr>
<tr>
<td>ESA</td>
<td>European Space Agency</td>
</tr>
<tr>
<td>GA</td>
<td>Grant Agreement</td>
</tr>
<tr>
<td>HMI</td>
<td>Human Machine Interface</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>IPR</td>
<td>Intellectual Property Rights</td>
</tr>
<tr>
<td>MPC</td>
<td>Minor Planet Center</td>
</tr>
<tr>
<td>MVP</td>
<td>Model View Presenter</td>
</tr>
<tr>
<td>NEA</td>
<td>Near Earth Asteroid</td>
</tr>
<tr>
<td>NEO</td>
<td>Near Earth Object</td>
</tr>
<tr>
<td>NEOCC</td>
<td>NEO Coordination Center</td>
</tr>
<tr>
<td>NEODyS</td>
<td>Near-Earth Objects Dynamic Site</td>
</tr>
<tr>
<td>PL</td>
<td>Priority List</td>
</tr>
<tr>
<td>P3L</td>
<td>Physical Properties Priority List</td>
</tr>
<tr>
<td>RD</td>
<td>Reference Document</td>
</tr>
<tr>
<td>SCN</td>
<td>Spaceguard Central Node</td>
</tr>
<tr>
<td>SLA</td>
<td>Service Level Agreement</td>
</tr>
<tr>
<td>SOAP</td>
<td>Simple Object Access Protocol</td>
</tr>
<tr>
<td>SSA</td>
<td>Space Situational Awareness</td>
</tr>
<tr>
<td>SW</td>
<td>Software</td>
</tr>
<tr>
<td>TBC</td>
<td>To Be Completed</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
</tbody>
</table>
1.3 Applicable Documents


1.4 Reference Documents


[RD6] ESA NEO Coordination Centre http://neo.ssa.esa.int


1.5 Standards


[STD2] Web Service Description Language, v2.0 (http://www.w3.org/TR/wSDL20/).
2 System Overview

2.1 Project background

Impacts of near-Earth objects (NEOs) have contributed to mass extinctions and evolution, and it is a proven fact that NEOs will continue to hit the Earth at irregular intervals in the future, with the potential for catastrophic damage to life and property.

In the context of the NEOShield-2 project, astronomical observations of NEOs will be carried out to improve our understanding of their physical properties, concentrating on the smaller sizes of most concern for mitigation purposes, and to identify further objects suitable for missions for physical characterisation, and NEO deflection demonstration. The aim of NEOShield-2 is to expand and enhance the space-mission NEO target database to include targets of interest for NEO exploratory missions in addition to mitigation demonstration missions. The physical properties and measurement uncertainties, or quality flags, relevant to reconnaissance, sample-return and mitigation demonstration missions will be included.

The specialized potential-target database will initially be developed separately, and then connected to the NEOShield-2 NEO physical properties open data repository thus creating a dynamical web interface to allow the public user to search on NEO physical properties relevant to exploration or mitigation demonstration missions. Appropriate settings of criteria will enable required prioritised lists of potential NEO mission targets to be obtained.

This will eventually allow the onset of a “two-way” operational interface with the ESA NEO Coordination Centre (see section 2.2.1) for focusing physical observations to potential mission targets and for providing the information needed to properly plan and execute them successfully (e.g. assign priorities based on accessibility considerations, send/receive astrometric alerts, provide astrometry of challenging objects, prompt newly discovered object observation opportunities, answer to requests from the observers, etc.).

The NEOShield-2 NEO physical properties database and the dynamical web interface will be publicly accessible through the NEOShield-2 NEO Properties Portal, as shown in the context diagram of Figure 1. In this diagram the solid arrows represent the flux of data either generated or disseminated through the NEO properties portal, whereas the dotted lines represent the links with functions developed outside WP 11.3 and WP 9.1 or accessible outside the NEOShield-2 project (e.g. ESA NEOCC, see section 2.2.1).
The public availability of the NEOCC physical properties database allows to avoid developing redundant functionalities within the NEOShield-2 project. The data produced by the NEOShield-2 observational activity, once published on the NEOShield-2 portal, will be integrated into the NEOCC database through the regular updating procedures provided to ESA by DLR under SLA.

2.2 NEO Physical Properties Database and Dynamical Web Interface

The aim of the NEOShield-2 NEO Physical Properties Database and Dynamical Web Interface implementation is to coordinate observations, ease the identification of suitable mitigation or exploration mission targets and to be able to compute and visualize example mission scenarios under different assumptions, allowing the retrieval and display of data useful for:

- Planning observations
- Performing off-line scientific analysis
- Update physical properties
- Summarize mission profiles

One of the main targets of the new Dynamical Web Interface is to contribute to increase the fraction of NEO population with known physical characteristics via dedicated observation campaign as well as, consolidating the NEO physical properties and their accessibility via tables and plots.

In order to do so the following data will be used:

- the orbital data of the whole NEO population as provided by the NEOCC continuously updated catalogue;
- the ephemeris of the individual objects as described in [RD7];
- the known NEO physical properties available from the NEOCC Physical Properties Database.

As discussed in section 2.1 in order to avoid redundancies the NEOShield-2 NEO Physical Properties Database (see Fig. 1) shall include only objects observed during NEOShield-2...
campaigns. For each of them it will provide the data products at different processing levels (raw, intermediate and final) shall be stored.

The basic structure of a NEOShield-2 Dynamical Web Interface consists of a web page containing updated tables and plots of observed objects and potential targets from where shall be possible to locate and download related data files, the following data shall be available:

- For each object observed by NEOShield-2:
  - Images (spectra, light curves).
  - Data files (in two-column format: e.g. photometry vs time).
  - Final products (e.g. size, spectral type etc).
- The information produced by SW tools:
  - Mission scenarios (table).
  - Observational support priorities (table).
  - Accessibility diagrams (plot).
- Observational support priorities:
  - For each NEO computes visibility-related quantities.
  - Generates a table of observable objects prioritised in terms of importance and urgency.
  - The priority table will be run daily through an automatic procedure and displayed on the Dynamic Web Interface.
  - The priority table can be run off-line for an arbitrary date.
- Accessibility diagrams:
  - For each NEO computes the delta-V corresponding to a best case Hohmann-like transfer trajectory.
  - Generates an accessibility diagram (H-plot) of the NEO population.
  - The plot can be customized by highlighting subpopulations (e.g. objects belonging to a certain spectral type, objects observed by NEOShield-2 etc.).

### 2.2.1 The ESA NEO Coordination Centre Physical Properties database

When the whole NEO population is involved the Physical Properties database publicly available through the ESA SSA-NEO Coordination Centre web portal (see [RD6]) should be addressed.

Developed within the framework of the Space Situational Awareness programme of the European Space Agency, the NEOCC web portal ([neo.ssa.esa.int](http://neo.ssa.esa.int)) hosts the EARN data (European Asteroids Research Node, [earn.dlr.de](http://earn.dlr.de)) thus representing a primary source of NEO physical properties. EARN contains up-to-date published data on all know NEAs and the corresponding bibliographic references; it is maintained by DLR Institute of Planetary Research in Berlin and it is considered the most detailed open data repository for physical and dynamical properties of NEOs.

EARN data have been integrated into the NEOCC system as a fully searchable database: a single query interface allows to display both the dynamical and the physical properties of any given NEOs or to search for objects within certain parameters range for further investigation. The physical properties data can be browsed from the NEOCC web portal by selecting the "search for objects" main menu option. Once the corresponding page is displayed (Figure 2) one can either enter an object name and being led to the corresponding summary page (Figure 3) or perform an “advanced search” among the dynamical and physical data stored in the NEOCC database (Figure 4).

The possibility of ingesting additional data provided by the NEOShield-2 project and of providing a direct access to the NEOCC search engine from the NEOShield-2 Properties Portal will be proposed as part of the NEOCC evolutionary maintenance.
Figure 2: The "search for objects" page of the NEOCC web site

Figure 3: The physical properties summary page of the NEOCC web site
Figure 4: The advanced search option of the NEOCC web site
3 Overall Architecture

The objective of the NEO Physical Properties Database and Dynamical Web Interface architecture is the definition of a modular and scalable system composed by independent components that interact between each other using well-defined interfaces.

The Dynamical Web Interface will be publicly available, providing the necessary tools for searching and visualizing NEOShield-2 data. Thus in what follows the users of the NEO Properties Portal will be referred to as NEOShield-2 users. The only requirements for running the tool on any location are:

- To have installed a web browser.
- To have access to the Internet.

The Figure 5 below shows the context diagram of the NEO Physical Properties Database and Dynamical Web Interface, the default flow of events would be the following:

1. A NEOShield-2 user launches a web browser to connect remotely through the Internet to the Dynamical Web Interface home page.
2. The Dynamical Web Interface is installed at DEIMOS premises infrastructure outside and accessible from the Internet.
3. Once the Dynamical Web Interface is launched, the user can perform a set of operations like the NEO search, visualization of tabular data, plotting display etc...
4. When accessing the Dynamical Web Interface it interfaces with its backend infrastructure located within the DMZ. All requests between the frontend and backend parts of the system use the standard HTTP over SOAP protocol that ensures a well-defined communication based on XML messages.

![Figure 5: NEO Physical Properties Database and Dynamical Web Interface context diagram](image)

From the design point of view, we can state the following high-level separation of capabilities:

- Logic related to the frontend presentation layer of the tool such as rendering of NEO data, tabular display, plotting etc...
- Logic related to the backend business logic layer of the tool that will interface with the OpenData Repository and the NEO Physical Properties Database.

Therefore a Three-Tier architecture shown in Figure 6 below has been selected as the best approach for the development of the Dynamical Web Interface:
The **Presentation** layer contains all client-side modules related with the query building forms and the results display of NEO data and related metadata in tabular or plot format.

The **Business Logic** layer is the most important part in the whole architecture because it defines the real-world business rules that determine how data can be created, displayed, stored, and changed. It works as a bridge between the Presentation and Data Access layers, all the user values received from the input web forms are being passed to it and in turn, the results received from the storage infrastructure in row data format are returned as suitable objects that can be easily displayed to the user.

The **Data Access** layer builds the necessary queries based on the received parameters from the Business Logic Layer and executes them over the NEO Physical Properties Database and the OpenData Repository, returning the results back.

The following diagram in Figure 7 displays the general structure of the components necessary to deliver data among the system; the dataflow among components is represented using arrows:
The dataflow within the system is as follows:

1. Observers upload their raw data which is composed of Images (spectra and light curves) and data files in two-column format to the OpenData Repository using FTP.
2. This raw data is processed by a set of offline scripts that insert the corresponding physical properties and final products into the NEO Physical Properties Database.
3. Once the products are available, NEOShield-2 end users can access the Dynamical Web Interface via HTTP and perform queries using its graphical interface.
4. The queries are processed by the Model Renderer and sent via SOAP over HTTP to the backend interface that translates them into SQL statements which are executed in the NEO Physical Properties Database.
5. Query results are propagated back to the interfaces and the Model Renderer that using the UI Builder presents the user the proper tables and plots.

In the following sections it is detailed the design approach of the different system components.

### 3.1 Data Model

#### 3.1.1 OpenData Repository

The OpenData Repository is an FTP server that provides public access to NEOShield-2 observers so they can upload raw data composed of images (spectra and light curves) and data files (in two-column format). The proposed structure is the one described in figure below:
The logic behind this structure is the following one:

- There will be a root folder called 'NEOShield' that shall contain one folder per telescope, like for example "NTT", "TNG" etc...
- Inside each telescope there will be the list of objects observed by that telescope, one folder per object.
- For each object observed there shall be one entry per observation date range and inside those observations campaigns folders the observers shall upload their images and ASCII data files in the dedicated folders.

### 3.1.1.1 Procedure to connect remotely to the OpenData Repository and upload files

The following steps must be executed to connect to the OpenData repository from any Internet location and upload images and data files associated to a given object:

1. Open a web browser and access DEIMOS VPN secure page:
   
   https://vpn.deimos-space.com

2. Once the page is loaded, use the NEOShield OpenData Repository credentials for public access:
   
   Name: [vpn_name]
   Password: [vpn_passwd]

3. Once the user is logged in, he/she shall be prompted to install the VPN web plugin to access NEOShield-2 Backend secured infrastructure.
4. Re-start the web browser and repeat the login process, once in the user shall press the button "Connect".
5. If the connection goes well then the secured access is granted and the user can access the OpenData Repository using a SFTP client tool with the following connection details:
   
   Host: [server-ip]
   Protocol: SFTP (port 22)
   User: [username]
   Pass: [passwd]

6. The structure of the repository shall be the same as described in Figure 8, if the user wants to upload data from a telescope which has no folder then he/she can create it and inside it create the proper object and date range folders for storing the files according to the convention defined in Figure 8.
3.1.2 NEO Physical Properties Database

The NEO Physical Properties Database shall store a consolidated version of the characteristics from objects observed during NEOShield-2 campaigns. Figure 9 shows the proposed table structure for the NEO Physical Properties Database in terms of fields and data types:
Figure 9: NEO Physical Properties Database proposed structure
3.2 SOAP Middleware Interface

3.2.1 JAX-WS SOAP API usage

The business layer providing the lower level access to the NEO Physical Properties database and generation of the information to be shown in the Dynamical Web Interface shall be composed of a set of Java classes that process input queries coming from the Dynamical Web Interface via SOAP over HTTP and translate them into SQL statements which are executed in the NEO Physical Properties Database.

3.2.2 Data Access Object usage

Access to persistent storage, such as to a database, varies greatly depending on the type of storage (relational databases, object-oriented databases, flat files, and so forth) and the vendor implementation, for that reason the SOAP Middleware Interface shall implement the Data Access Object (DAO) pattern for accessing to abstract and encapsulate all access to the NEO Physical Properties Database, working as an adapter between the component and the data source.

The business component that relies on the DAO uses the simpler interface exposed by the DAO for its clients. The DAO completely hides the data source implementation details from its clients which communicate via the Transfer Object. Because the interface exposed by the DAO to clients does not change when the underlying data source implementation changes, this pattern allows the DAO to adapt to different storage schemes without affecting its clients or business components.

Figure below shows the implementation of this pattern for the use case of the Celestial Body entity which has been characterized just with the "designator" attribute for simplicity:

1. The "CelestialBody" JavaBean entity contains the definition of all celestial bodies' attributes (like the "designator" field). These entity beans are transactional objects representing persistent data and expose the values of attributes by providing accessor methods (setters and getters) for each attribute it wishes to expose.
2. The interface class "IPhysicalPropertiesDAO" encapsulate the signature methods for accessing the different Physical Properties JavaBean classes, like the "CelestialBody" JavaBean.
3. The "PhysicalPropertiesDAOImpl" class implements the methods provided in the interface with the proper business logic inside.
4. The "PhysicalPropertiesTransferObject" class is used to encapsulate the business data coming from the client. A single method call is used to send and retrieve the Transfer Object. When the client requests the enterprise bean for the business data, the enterprise bean can construct the Transfer Object, populate it with its attribute values, and pass it by value to the client.
3.3 Dynamical Web Interface

For details about the Dynamical Web Interface architectural design see [RD8].

3.4 Architectural Open Points

The following issues remain open in the design of the NEO Physical Properties Database:

<table>
<thead>
<tr>
<th>Open Point ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[OP-1]</td>
<td>A clear data policy must still be established (e.g. before observations reduction and analysis no data can be made publicly available).</td>
</tr>
<tr>
<td>[OP-2]</td>
<td>The number of objects to be observed during the NEOShield-2 project must be addressed; We have made a rough estimation of 500 objects.</td>
</tr>
<tr>
<td>[OP-3]</td>
<td>It has to be clarified what are the different data types that you expect us to store (e.g. images, photometry files, spectroscopy, plots etc.) and the corresponding size for each object observed; We have estimated one image and a couple of ASCII files per object.</td>
</tr>
<tr>
<td>[OP-4]</td>
<td>It has to be clarified which facilities are likely to carry out most of the observations (e.g. NTT, TNG etc).</td>
</tr>
<tr>
<td>[OP-5]</td>
<td>Regarding the OpenData Repository, clarify if there are any preferences in organizing the data for subsequent retrieval; One single public user for all observers or different dedicated users?</td>
</tr>
<tr>
<td>[OP-6]</td>
<td>It has to be clarified if there is any baseline Operating System to be used across the project. SUSE Linux Enterprise Server 11 SP1 (x86_64) is the selected option currently installed @DEIMOS</td>
</tr>
</tbody>
</table>

Table 1: Architectural open points in the implementation of the NEO Physical Properties Database
## Implementation Technologies

The following table shows the third party products used in the implementation of the NEO Physical Properties Database and Dynamical Web Interface and the corresponding IPR information.

<table>
<thead>
<tr>
<th>Product</th>
<th>Vendor Name</th>
<th>Version and License</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUSE Linux Enterprise (operating system)</td>
<td>SUSE Linux (<a href="http://www.suse.com/">http://www.suse.com/</a>)</td>
<td>SLES11 Service Pack 1 x86_64 (kernel release: 2.6.32.19-0.3) License at <a href="http://www.novell.com/company/legal/">http://www.novell.com/company/legal/</a></td>
</tr>
<tr>
<td>Java JDK/JRE (J2EE VM)</td>
<td>Oracle Corporation (<a href="http://www.oracle.com/">http://www.oracle.com/</a>)</td>
<td>v1.7.0_60 SCSL (<a href="http://www.oracle.com/technetwork/java/scsl-1-1-149938.txt">http://www.oracle.com/technetwork/java/scsl-1-1-149938.txt</a>)</td>
</tr>
<tr>
<td>MySQL Database (relational database)</td>
<td>Oracle Corporation (<a href="https://www.mysql.com/">https://www.mysql.com/</a>)</td>
<td>v5.5.24 SCSL (<a href="http://www.oracle.com/technetwork/java/scsl-1-1-149938.txt">http://www.oracle.com/technetwork/java/scsl-1-1-149938.txt</a>)</td>
</tr>
<tr>
<td>Apache Tomcat (J2EE container)</td>
<td>Apache Software Foundation (<a href="http://tomcat.apache.org/download-70.cgi">http://tomcat.apache.org/download-70.cgi</a>)</td>
<td>v7.0.62 Apache License v2.0 (<a href="http://www.apache.org/licenses/LICENSE-2.0.html">http://www.apache.org/licenses/LICENSE-2.0.html</a>)</td>
</tr>
<tr>
<td>Apache Maven (automation tool)</td>
<td>Apache Software Foundation (<a href="http://ant.apache.org/">http://ant.apache.org/</a>)</td>
<td>v3.2.1 Apache License v2.0 (<a href="http://www.apache.org/licenses/LICENSE-2.0.html">http://www.apache.org/licenses/LICENSE-2.0.html</a>)</td>
</tr>
</tbody>
</table>

Table 2: Third party products used in implementation of the NEO Physical Properties Database
Appendix A: Installation procedure

This appendix describes procedure to install the NEO Physical Properties Database and Dynamical Web Interface.

4.1 Installation Pre-requisites

This section details the Third-Party components that should be present prior to installing the NEO Physical Properties Database and Dynamical Web Interface in the target (development or operational) machine. The ‘SRC_ROOT’ path refers to the path in the NEOShield-2 DEIMOS ‘trunk’ code repository where the selected items are located.

4.1.1 Operating System

The reference Operating System for the project is SuSE Linux Enterprise Server v11 SP1. For checking the operating system, execute the following command on a terminal:

```
$ cat /etc/issue
Welcome to SUSE Linux Enterprise Server 11 SP1 (x86_64) - Kernel \r \t1.
```

4.1.2 Java VM

The Java reference version to be used is JRE 7; for installing the Java virtual machine execute the following commands:

1. Copy the installation bundle and unpack it in the desired location:

   ```
   $ cp SRC_ROOT/COTS/Java/jdk-7u60-linux-x64.tar $HOME
   $ cd
   $ tar xfvz jdk-7u60-linux-x64.tar
   ```

2. To be able to run Java from any location in the build platform we must add the path to Java executable binary to the $PATH variable:

   ```
   $ export JAVA_HOME=$HOME/jdk1.7.0_60/bin
   $ export PATH=..:$JAVA_HOME/bin:$PATH
   ```

3. For checking the Java version installed run:

   ```
   $ java -version
   java version "1.7.0_60"
   Java(TM) SE Runtime Environment (build 1.7.0_60-b19)
   Java HotSpot(TM) 64-Bit Server VM (build 24.60-b09, mixed mode)
   ```

4.1.3 Apache Maven

For building the Dynamical Web Interface SOAP services we have used Maven automation tool, to install Maven follow these steps:
1. Copy the installation bundle and unpack it in the desired location:

```
$ cp SRC_ROOT/COTS/Maven/apache-maven-3.2.1-bin.tar.gz $HOME
$ cd
$ tar xfvz apache-maven-3.2.1-bin.tar.gz
```

2. To be able to run Maven from any location in the build platform we must add the path to Maven executable binary to the $PATH variable:

```
$ export MAVEN_HOME=$HOME/apache-maven-3.2.1
$ export PATH=.:$MAVEN_HOME/bin:$PATH
```

3. Maven repository dependencies are available in the '$HOME/.m2/repository' folder.

4. Notes on Maven execution flags:
   - If we need to include debugging details during compilation then we can run the same Maven command but adding the '-X' flag. In this installation procedure we have not included this flag for the sake of simplicity.
   - If we need to make an offline build of the services then we can run the same Maven command but adding the '-o' flag. Maven will not attempt to connect to any remote repository to retrieve artefacts and shall get the necessary dependencies from the local repository available in the '.m2/repository' folder. In this installation procedure we have included this flag to force the offline compilation mode.

4.1.4 MySQL Relational Database

This procedure installs the MySQL Server 5.6 using RPM binary bundles:

1. Copy the installation bundles to the deployment server:

```
$ cp SRC_ROOT/COTS/MySQL/MySQL-server-5.6.23-1.sles11.x86_64.rpm $HOME
$ cp SRC_ROOT/COTS/MySQL/MySQL-client-5.6.23-1.sles11.x86_64.rpm $HOME
```

2. As 'root' user install the packages:

```
$ sudo rpm -ivh MySQL-server-5.6.23-1.sles11.x86_64.rpm
$ rpm -ivh MySQL-client-5.6.23-1.sles11.x86_64.rpm
```

3. Modify MySQL root user password. You must set the password to the root user of MySQL environment. The password must be "neoroot":

```
$ mysqladmin -u root password [root_passwd]
```

4. Set MySQL as no case sensitive. By default, MySQL Server Linux version is case sensitive. We must change this property to improve server performance. Edit the MySQL config file:

```
$ vi /etc/my.cnf
```

And add the following entries:
5. Also allow remote connections to the Database by commenting the following line:

```bash
#bind-address = 127.0.0.1
```

6. Restart MySQL Service. To do this type:

```bash
$ sudo /etc/init.d/mysql stop
$ sudo /etc/init.d/mysql start
```

7. Exit the root shell.

8. To connect to the database and display the available databases simply execute:

```bash
$ mysql -h localhost -u root -p (password=[root_passwd])
```

4.1.5 Apache Tomcat

The Apache Tomcat reference version to be used is 7.0.62; for installing Tomcat engine execute the following commands:

1. Copy the installation bundle and unpack it in the desired location:

```bash
$ cp SRC_ROOT/COTS/Tomcat/apache-tomcat-7.0.62.tar.gz $HOME
$ cd
$ tar xfvz apache-tomcat-7.0.62.tar.gz
```

2. To be able to run Tomcat we must add the path to its installation root folder to the $PATH variable:

```bash
$ export CATALINA_HOME=$HOME/apache-tomcat-7.0.62
$ export PATH=.:$CATALINA_HOME/bin:$PATH
```

3. For starting Tomcat simply execute:

```bash
$ cd $CATALINA_HOME/bin
$ ./startup
```

4. Tomcat’s welcome page should be now available at the following url:

```
thttp://172.23.5.230:8080/
```
5. To stop Tomcat simply execute:

   $ cd $CATALINA_HOME/bin
   $ ./shutdown

4.2 Build procedure from sources

4.2.1 Dynamical Web Interface

4.2.1.1 Eclipse IDE Installation

For building all DPC HMI components we shall use the Eclipse IDE version "Luna" SR2 with SDK 4.4 bundle, the following steps detail the installation in the development server:

1. Get the Eclipse Luna SR2 with SDK 4.4 bundle from the source repository which is available in folder:

   SRC_ROOT/COTS/Eclipse/eclipse-jeeyluna-SR2-linux-gtk-x86_64.tar.gz

   **Note:** For Windows platforms get instead the ‘eclipse-jeeyluna-SR2-win32-x86_64.zip’ bundle.

2. Copy the zipped file to your local drive and unpack it in the desired location, in this installation procedure we shall deploy in the user’s $HOME folder:

   $ mkdir $HOME/eclipse_4.4
   $ cd $HOME/eclipse_4.4
   $ cp SRC_ROOT/COTS/Eclipse/eclipse-jeeyluna-SR2-linux-gtk-x86_64.tar.gz.
$ tar xfvz eclipse-jee-luna-SR2-linux-gtk-x86_64.tar.gz

The following folder shall be created:

$ $HOME/eclipse_4.4/eclipse

3. Run the ‘$HOME/eclipse_4.4/eclipse/eclipse’ executable file to launch the tool, if the file has no execution permissions then assign them by doing:

   $ cd $HOME/eclipse_4.4/eclipse
   $ chmod +x eclipse
   $ ./eclipse &

4. The first time the application is launched it asks to setup the desired workspace location, by default we shall set this parameter to be:

   $HOME/workspace_4.4

Once the application is loaded the “Welcome page” is displayed, we can close it and the tool it is ready to be used as coding editor.

**Figure 12: Eclipse Welcome Page**

### 4.2.1.2 Google plugin for Eclipse Installation

To install the Google plugin for Eclipse follow these steps:

1. In Eclipse IDE, select the menu option "Help > Install New Software..." and in the dialog that appears, enter the following update site URL into the 'Work with' text box:

   [https://dl.google.com/eclipse/plugin/4.4](https://dl.google.com/eclipse/plugin/4.4)
2. Press the “Enter” key and then the list of available packages shall be displayed as shown below:

![Google plugin for Eclipse installation packages](image)

Figure 13: Google plugin for Eclipse installation packages

3. From the list of available packages, we shall install the following ones and click “Next”:
   - Google App Engine Tools for Maven
   - Google Plugin for Eclipse
   - GWT Designer for the Google Plugin for Eclipse
   - Google Web Toolkit SDK

4. Review the features that you are about to install, if everything is OK then click “Next”.
5. Read the license agreements and then select ‘I accept the terms of the license agreements’ and click ‘Finish’ to go on with the installation.

Figure 14: Google plugin for Eclipse installation packages summary
6. After the installation is completed, you will then be asked if you would like to restart Eclipse, click ‘Restart Now’ to complete the installation process.

### 4.2.1.3 Web Portal deployment

To build the Dynamical Web Interfaces and be able to get the proper *.war file ready to be deployed in Tomcat J2EE container follow these steps:

1. Import the DataAccessObjects project. In Eclipse IDE, select the menu option “File > Import... > SVN > Project from SVN” and create a new repository location with the url `https://sputnik.deimos-space.com/svn/neoshield-2/trunk/WebPortal/DataAccessObjects`.
2. Update Project. Right click in project and select “Maven > Update Project...”
3. Clean the project. Select the menu option “Project > Clean...”, the option “Project > Build Automatically” is checked.
4. Clean, compile and assembly the project with production profile in Maven. A file with an extension *.jar will be created in the project inside the target directory.
5. Import the project. In Eclipse IDE, select the menu option “File > Import… > SVN > Project from SVN” and create a new repository location with the url https://sputnik.deimos-space.com/svn/neoshield-2/trunk/WebPortal/Neoshield.

6. Update Project. Right click in project and select “Maven > Update Project…”

7. Clean the project. Select the menu option “Project > Clean…”, the option “Project > Build Automatically” is checked.

8. Maven clean. Execute with Maven –update-snapshots clean install –U -X:

9. GWT compile. Execute with maven gwt:compile:
10. Deploy with production profile. Execute `mvn tomcat7:deploy`. A file with an extension `.war` will be created in your project inside the target directory and it will deploy it automatically in the remote machine.

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
  http://maven.apache.org/xsd/settings-1.0.0.xsd">
  <servers>
    <server>
      <id>Tomcat</id>
    </server>
  </servers>
</settings>
```
4.2.1.1 Daemon installation

The daemon is continually running looking for new requests to generate a priority list zip file and send it by email to the respective user. To install it follow these steps:

1. Import the CallP3Lscript project. In Eclipse IDE, select the menu option "File > Import… > SVN > Project from SVN" and create a new repository location with the url https://sputnik.deimos-space.com/svn/neoshield-2/trunk/WebPortal/CallP3Lscript.
2. Update Project. Right click in project and select “Maven > Update Project…”
3. Clean the project. Select the menu option "Project > Clean…", the option "Project > Build Automatically" is checked.
4. Clean, compile and assembly the project with the production profile in Maven. A file with an extension *.jar will be created in the project inside the target directory.

![Maven clean compile assembly](image)

**Figure 32: Maven clean compile assembly**

5. Copy the jar file in the /home/openeo/P3L_Hplot_tools/P3L_ondemand directory.
6. Give execution permission to this file.
   chmod +x /home/openeo/P3L_Hplot_tools/P3L_ondemand/CallP3Lscript-0.1.jar
7. Copy the daemon script in the same directory.
8. Give execution permission to this file.
   chmod +x /home/openeo/P3L_Hplot_tools/P3L_ondemand/P3L_Hplot.sh
9. Add the task to the cron daemon.
   crontab -e
   @reboot /home/openeo/P3L_Hplot_tools/P3L_ondemand/P3L_Hplot.sh start 2>&1
4.3 Installation from binaries

4.3.1 NEO Physical Properties Database

The NEO Physical Properties Database shall be stored in the MySQL server deployed within the NEOShield virtual machine described in section 4.1.4. Once the MySQL server is installed, it is necessary to create the `neoshield` database and their associated tables using the following commands:

```bash
SRC_ROOT/Database/bin/neoshield_bbdd.sql

1. Create `neoshield` database:

   $ mysql --user=root --password=[root_passwd] -e "create database neoshield character set utf8;"

2. Create `neoshield` tables:

   $ mysql --user=root --password=[root_passwd] -A -Dneoshield < neoshield_bbdd.sql

3. For managing NEOShield tables, we shall create a dedicated user and provide the necessary permissions:

   $ CREATE USER 'neoshield'@'localhost' IDENTIFIED BY 'neoope';
   $ GRANT ALL PRIVILEGES ON * . * TO 'neoshield'@'localhost';
   $ FLUSH PRIVILEGES;

4. Connect as `neoshield` user to the database and display its tables:

   $ mysql -h localhost -u neoshield -p (password=[neoope_passwd])
   mysql> show databases;
   mysql> use neoshield;
   mysql> show tables;
```
4.3.2 Dynamical Web Interface

To deploy the Dynamical Web Interface into Tomcat J2EE container follow these steps:

1. Remove the settings.properties inside the project
2. Check apache tomcat is started by executing in the command prompt:

   $ ps -ef | grep tomcat

   If the output contains the whole path of the tomcat folder $CATALINA_HOME), then it is running.

3. Start apache tomcat if it is not running:

   $ cd $CATALINA_HOME/bin
   $ ./start

4. Deploy with production profile. Execute mvn tomcat7: deploy. A file with an extension *.war will be created in your project inside the target directory and it will deploy it automatically in the remote machine.
5. Copy the Neoshield.properties file. The Neoshield.properties is located inside the neshield project in the config folder. Copy it in the virtual machine in the desired path. The Neoshield.properties has the path of the Physical Properties Priority List, P3L ondemand, Accessibility Hplots, Observations Status Page and Open Data Repository Root directories.


7. Edit that path in settings.properties file located in `$CATALINA_HOME/webapps/Neoshield/WEB-INF/classes/com/dms/neoshield2/resources` with the path of the Neoshield.properties file.

8. Restart Tomcat.
   $ cd $CATALINA_HOME/bin
   $ ./shutdown.sh
   $ ./startup.sh

4.4 Confidence Test

To verify the installation has been performed rightly execute in the command prompt:

   $ cd $CATALINA_HOME/logs
   $ tail -f catalina.out
The user can observe an information message “Server startup in 2590 ms” without exceptions, therefore the application is correctly deployed and the user can start to navigate with the url:

http://172.23.5.230:8080/Neoshield
4.5 Maintenance Tasks

4.5.1 Logs

The logs of not only the dynamical web interface, but also the daemon are generated in the /home/openeo/apache-tomcat-7.0.62/logs path.

The level of both logs files can be configured in the log4j2.xml file. In the case of the dynamical web interface is generated in the /home/openeo/apache-tomcat-7.0.62/lib. On the other hand, in the daemon is located inside the CallP3Lscript project in the src/main/resources folder. The user can choose between three levels: DEBUG, INFO or ERROR.

4.5.2 Controlling the execution of the daemon

To see the status of the daemon run:

/home/openeo/P3L_Hplot_tools/P3L_ondemand/P3L_Hplot.sh status
If it is stopped, the user can start its execution with the command:
/home/openeo/P3L_Hplot_tools/P3L_ondemand/P3L_Hplot.sh start
On the other hand, it is running, the user can stop or restart it with the respective commands:
/home/openeo/P3L_Hplot_tools/P3L_ondemand/P3L_Hplot.sh stop
/home/openeo/P3L_Hplot_tools/P3L_ondemand/P3L_Hplot.sh restart