



# Dose Hunter User Guide

Oncopole Claudius Regaud, France

Created by: **Luc SIMON** (simon.luc (at) iuct-oncopole.fr)

Contributors: **Luc SIMON, Bradley BEEKSMA, François-Xavier ARNAUD, Killian LACAZE, Farzam SAYAH**

## Abstract

This document describes how to install and how to use DoseHunter, a tool to collect a large number of data in your Varian Eclipse database.

# 1. What is DoseHunter ?

**DoseHunter** is a stand-alone executable for Varian Eclipse V15.x. DoseHunter automatically collects dose data (dose max, dose min, D95%, etc.) for a large number of patients in your database. DoseHunter creates a .csv file with your data that can be easily treated with Excel, Python...

## 2. Installation

If you are lucky the executable application can be directly run on your Eclipse station (not a Citrix station): see part 2.1. However it may be necessary to compile the source code (part 2.2).

### 2.1. Executable

In an empty directory of your Varian Eclipse v15.x Station (tested with v15.6), download the 3 following files from github (<https://github.com/uhqd/DoseHunter>):

- DoseHunter.exe
- VMS.TPS.Common.Model.API.dll
- VMS.TPS.Common.Model.Types.dll

The 2 last files can be replaced by your own .dll corresponding to your version of Eclipse (available on any Eclipse station) at the following path (can be slightly different on your station):

C:\Program Files(x86)\Varian\RTM\15.6\esapi\API\VMS.TPS.Common.Model.API

C:\Program Files(x86)\Varian\RTM\15.6\esapi\API\VMS.TPS.Common.Model.Types

Copy them in your working directory. If DoseHunter doesn't work, you probably have to compile the source (see 2.2)

### 2.2. Compilation

Download the repository from github <https://github.com/uhqd/DoseHunter>

Open the solution (.sln) in Microsoft Visual Studio Community.

You must add in the reference list of the project, the 2 .dll les (see part 2.1 where to find these files). Compile and build the project in a directory. Copy the two .dll les in the same directory than the .exe le. This working directory must also contains the three user text files: id.txt, planfilter.txt and index.txt (see sections 3.1, 3.2 and 3.3). An example of these files can be found on the github repository (<https://github.com/uhqd/DoseHunter>).

## 3. Getting started

On a Eclipse station (not citrix) a simple double click on the DoseHunter.exe file executes the program.

The 5 following files must be placed in the same working directory

- **VMS.TPS.Common.Model.Types.dll**
- **VMS.TPS.Common.Model.API.dll**
  
- **id.txt**
- **index.txt**
- **planfilter.txt**

The Windows console displays details on the execution. At the end data can be found in **out/**

## 2.1. id.txt

id.txt is a simple text file (can be done with the Microsoft Notepad) that contains only the list of your patient IDs (one by line). It can be done by copy/paste a column from an Excel file in the Notepad. Do not add any space character or any additional line breaks. An exemple can be found on github.

## 2.2. index.txt

index.txt is another text file to indicate which data you want to collect for your list of patients (see previous section 2.1). Each line must be in the following format:

```
<struct. name 1>;<struct name 2 (opt)>;<struct. name 3 (opt)>,<index>,<index>,<index> (etc.)
```

Example:

```
Heart;HEART;theheart,max,min,median,D95%  
PTV,max,D95cc,median  
Liver;liver,max,V50cc,vol
```

Please, notice the difference between:

,      and      ;

Each line is separated in different elements using a comma (,). The first element is the structure name. It can be interesting (but not mandatory) to give different spelling for the same structure name. Then these different names must be separated by a semicolon (;) DoseHunter will search the first spelling of the structure name in the patient structure set. If (and only if) DoseHunter doesn't find this structure name, it will try the second one, the third one... In our example, the first structure can be named **Heart**, **HEART** or **heart**. If, for a patient, the structure Heart exists (preferred spelling), other spelling will be ignored. DoseHunter is case insensitive, thus the 2 first spelling (Heart and HEART) are redondant

The other elements on the line are the indexes that you want to collect for a given structure. You can add any number of indexes. Candidates are for a given structure:

- **vol**: volume (cc)
- **min**: dose minimum (Gy)
- **max**: dose maximum (Gy)
- **mean**: mean dose (Gy)
- **median**: median dose
- **DXX%** or **DXXcc**: e.g. D95% or D2.5cc : Dose (Gy) received by 95% or 2.55 cc of the structure
- **VXX%** or **VXXcc**: e.g. V49.6% or V49.6cc : Volume (% or cc) that received 49.6 Gy
- **HI** or **hi** : Homogeneity Index in the structure:

$$HI = \frac{D2-D98}{D50}$$

where D2, D50 and D98 are the doses received by 2%, 50% and 98% of the chosen structure, respectively.

- **GI** : gradient index (does not depend on the structure) ; Volume of BODY receiving 50% divided by Volume of Body receiving 100% of plan Total Dose
- **CIxx**: e.g. CI95 : Conformity Index for 95 isodose

CI95 = PIV / TV where PIV is the volume of isodose 95% and TV the structure volume.

- **PIxx** : e.g. PI95 : Paddick Conformity Index for the isodose 95%

$$PI95 = \frac{TV_{PIV}^2}{TV \times PIV}$$

where  $TV_{PIV}$  is the structure volume receiving 95% of total dose, PIV is the volume of the isodose 95% and TV the structure volume.

Please notice that all doses are absolute.

## 2.3. planfilter.txt

It may be convenient to choose which plans of your patient you want to explore. For example, you may need to exclude QA plans. It is possible to filter the plans by using the planfilter.txt file. If the file is not in the working directory, default values will be used. The file must not be modified, except the values after the ":". This is an example of the file content:

```
# HOW TO FILTER YOUR PLAN
# DO NOT MODIFY THIS FILE EXCEPT THE PART AFTER THE ":"
# keep only the plans with a total dose > to a value
```

```

Min Total Dose (Gy):60.0
# keep only the plans with a total dose < to a value
Max Total Dose (Gy):200.0
# Treat approved? If "no" treat approved plans will be excluded.
TreatApproved plan:no
# Planning approved? If "no" planning approved plans will be excluded.
PlanningApproved plan:no
# Unapproved? If "no" unapproved plans will be excluded.
Unapproved plan:yes
# Named plans? If "no", plans with a name will be excluded.
Named plan:yes
# Unnamed plans? If "no", plans with no name will be excluded.
Unnamed plan:yes
# keep it if it contains a string? If "yes", plans will be kept only
if they contain the string
Plan name must contain a string:no:toto
# Exclude it if it contains a string? If "yes", plans will refused
if they contain the string
Exclude if plan name contains:no:toto
# Explore Sum plans ? if no, sum plans are ignored, yes to explore them
Explore Sumplans:yes
# Explore Uncertainty plans ? if no, uncertainty plans are ignored, yes to explore them.
Warning: it slows the execution
Explore uncertainty:no
# keep the COURSE it if it contains a string? If "yes", course will be kept only if they
contain the string
Course name must contain a string:yes:tutu

```

In this example, the user wants to keep only the following plans:

- Total dose between 60 and 200 Gy
- exclusion if plan status is "Treat Approve"
- exclusion if plan status is "Planning Approve"
- keep plans if plan status is "Unapproved"
- keep plans with a name (All plans have an ID but not necessarily a name)
- keep plans without a name
- no test on the ID here. The user could choose to keep or exclude plans with an ID that contains a string, e.g. if you want to exclude all plans for which the ID contains the string 'breast' or 'BREAST' the last line of the file must be:

```
Exclude if plan name contains:yes:DTO:dto
```

- Sum plans are not ignored. Please notice that if Sum plans are explored, all filters will be ignored for these plans except "test if the plan has a name", "keep it if the plan name contains a string", "exclude it if the plan name contains a string". Moreover, the plan sum has no creation date (dose hunter will give the date of the last plan) and creation user will be set to plan sum. The number of fractions is the sum of all fractions of all plans. The Total dose is the sum of the total doses of all plans.

- uncertainties plans will not be analyzed. Warning: analysis of plan uncertainties slows the execution a lot. Resolution can be changed by developers in the code in the class DVH by changing the value 0.001 in the following lines:

```

...
return uncert_plan.GetDVHCumulativeData(structure,
DoseValuePresentation.Absolute,
VolumePresentation.Relative, 0.001);

```

```
...
return uncert_plan.GetDVHCumulativeData(structure,
DoseValuePresentation.Absolute,
VolumePresentation.AbsoluteCm3, 0.001);
```

- Finally, in the previous example, only course with a courseID containing the string 'tutu' will be analyzed

## 2.4. Exécution of DoseHunter

When double-clicking on DoseHunter.exe, the patient list is read (see the description of id.txt). Every course is opened and for each course, every plan is opened (except is the plan is refused, due to the plan filters, see section 2.3).

Then, for each patient of the list, DoseHunter will try to collect the chosen data (see part 2.2) and to write them in an output file (.csv).

No need to open Eclipse on the Station, DoseHunter is a stand alone executable!

When a plan is opened, its name and its total dose is displayed on the console. Then DoseHunter get the DVH data and collects the chosen ones. If the structure is not found the data is not reported but the program continues to run.

```

C:\Users\simon_lu\Desktop\dose_hunter\DoseHunter.exe

DOSE HUNTER
Luc SIMON. 2021

Plans filters. The following plans will be used to select the plans:
Total dose between 60.00 and 200.00 Gy
Keep planning approved plans? False
Keep treatment approved plans? False
Keep unapproved plans? True
Keep plans containing a particular string? False
Exclude plans containing a particular string? False

Directory ./out deleted and recreated...
1 Patient: [REDACTED] (201905875)
Plan: DT0-SEINGGGBL
Total dose = 66.994 Gy
Coeur found
Canal med found
PTU cni found
Cannot find the structure cani with this name or other names
For this patient 1/1 accepted plans

2 Patient: [REDACTED] (202012911)
Plan: SeinGgg
Total dose = 67.564 Gy
Coeur found
Canal med found
PTU cni found
Cannot find the structure cani with this name or other names
For this patient 1/1 accepted plans

Number of accepted/total patients: 2/2 (accepted : at least one accepted plan)
Number of accepted/total plans: 2/2
Please type Enter
```

## 3. Output data

An output/ directory is created for the output files. Be careful, this directory is deleted and recreated at each execution of DoseHunter. At the end of the execution the console displays a synthetic report of the execution and waits for the user to type ENTER.

The output/ directory contains 2 files:

- **log.txt** that contains more or less the same messages that have been displayed on the console during the execution of DoseHunter.

- data.csv that can be opened with Excel or a Python script and that contains the collected data. Each data is separated by a semicolon (;) and each line has the following format (one line per accepted plan, a patient can have several accepted plans):

```
<patient ID>;<course ID>;<plan ID>;<plan date>;<author of the plan>;
<total dose>;<dose per fraction>;<number of fractions>;<MU>;<MI (MU/fraction)>;
<plan normalisation value>;<index>;<index>;<index>...
```

The first line of the .csv file gives the titles of the columns. The other lines give the data separated by a semicolon. If the value of an index is not found (e.g. DoseHunter did not find a structure) the cell is left empty (it will appear in the csv file as a double semicolon, e.g. 3;2;4;;1).