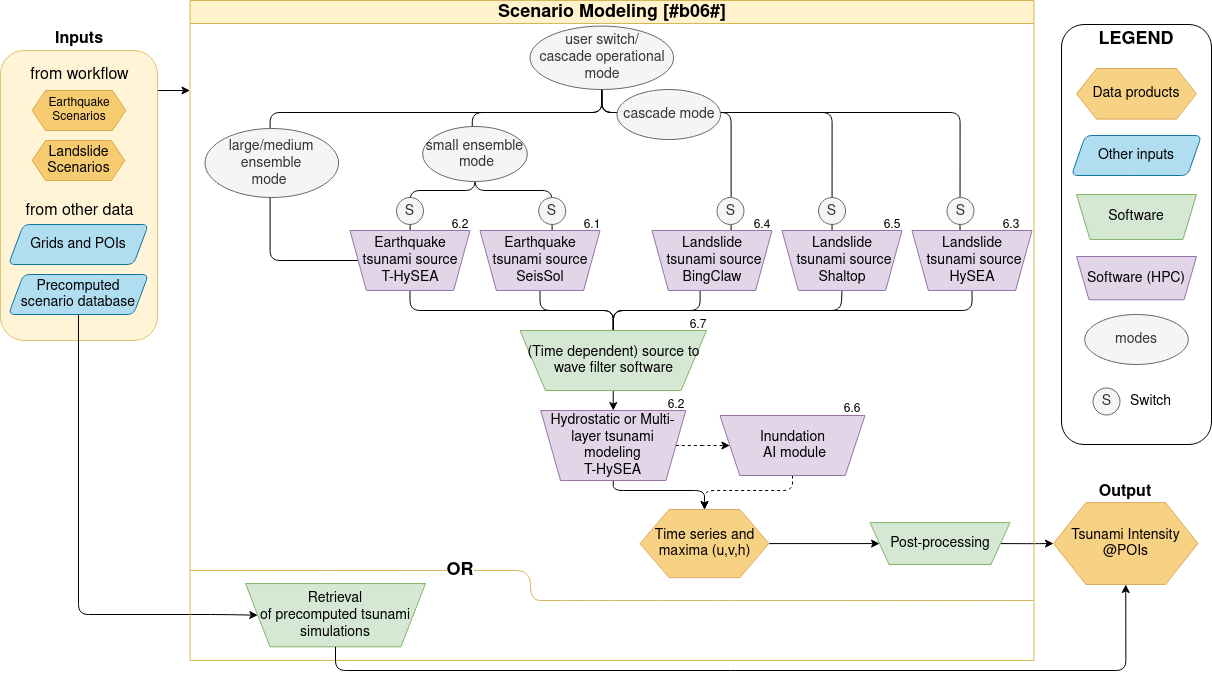
Interface Module - Documentation

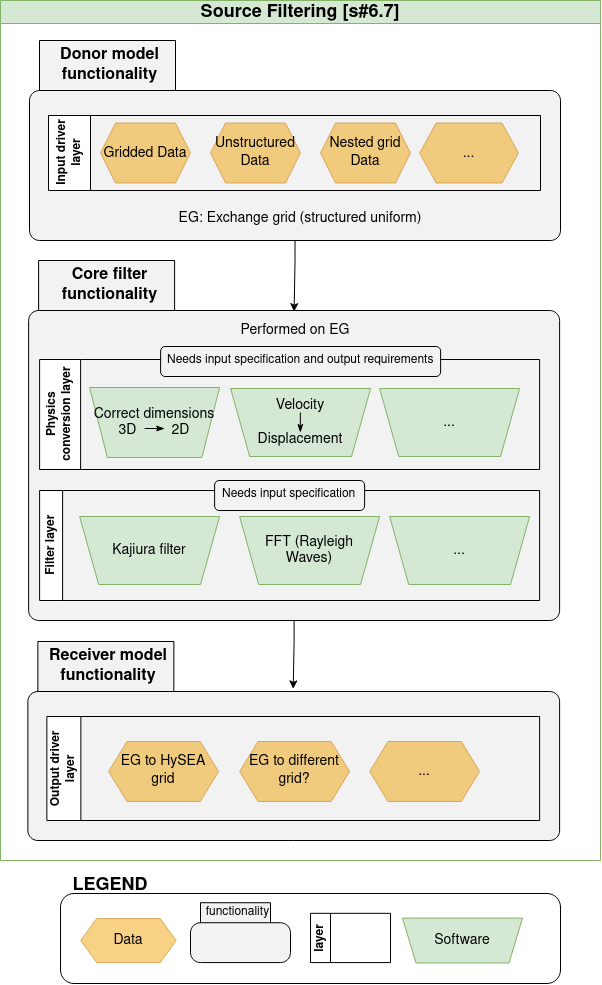
# Introduction

This document contains the description of WP6’s interface module (or source2wave filter) that is used within the PTF workflow. Within the PTF workflow, in particular the PTF component #b06# (Scenario Modeling), different source models can be used to provide source data for the tsunami generation. The following figure shows an outline how this is performed within the workflow. Possible source models include earthquake tsunami sources (T-HySEA and SeisSol) and landslide tsunami sources (BingClaw, Shaltop, landslide HySEA).



# Conceptual idea

A need for an interface module arises since the source models do not necessarily produce outputs in the (data) formats that are required by the receiver model (T-HySEA). Consequently, in a general way, we need to consider the data provided by the source models.   
Since those models “donate” the source information, we call this level the donor model level. On the donor model level, the source data has to be read, possibly interpolated to a specified grid and saved.   
Throughout the interface module, we need to somehow convert the data to the data format that is required by the receiver model (T-HySEA). For this, a new (intermediate) layer has to be introduced where all manipulations are performed on the donor model data. This level is called the exchange grid level and on this level, all the core filtering functionalities are applied. This includes filtering functionalities (such as the Kajiura filter), or routines to correct dimension or physical unit conversion.

The final level is called the receiver model level and it serves to correctly write the data so that it can be used by the receiver model (T-HySEA). The following figure outlines the idea of this conceptual idea.

# Implementation and use

The implementation of the interface model is written in python and can be found on the WP6 GitHub page ([Link to module](https://github.com/dtgeoeu-wp6-tsunamis/Interface-module)). While the Readme on the interface module’s GitHub repository explains the usage, this document will also outline how the interface module needs to be used.

To run the interface module, the following Python modules have to be installed:

Standard Packages:

* argparse
* datetime
* math
* numpy
* os
* pathlib
* pyproj
* scipy
* time

Non-standard packages:

* netCDF4
* pyvista
* seissolxdmf
* vtk

Once all the Python modules have been installed, the script can be run on a terminal with the command:  
  
python interface\_module.py --donor donor\_model --CRS\_reference lon, lat donor\_output bathy\_file resolution (--receiver receiver\_model --filter filter --casename casename)

where the following arguments are required:

* –donor\_model donor\_model: donor\_model can be ‘bingclaw’, ‘seissol’ or ‘shaltop’
* –CRS\_reference lon, lat: longitude and latitude coordinates of the exchange that will be used/defined within the module. This corresponds to the origin of the mesh (= lower left corner)
* donor\_output: path where the output of the donor model is stored (for BingClaw or Shaltop) or output file (for SeisSol)
* bathy\_file: bathymetry file of the computational domain that has to be provided along with the donor model data
* resolution: spatial resolution (in both x/longitude and y/latitude direction) that will define the exchange grid. Has to be provided in meters.
* –receiver receiver\_model: specifier for the receiver model (currently only ‘hysea’ is possible)
* –filter filter: handle to choose a filter (‘none’ or ‘kajiura’, all other cases will be defaulted to ‘none’)
* –casename casename: prefix that will be used for the output file name(s)

The arguments in parentheses are optional. It should be noted that the cases have to be given as written above (all lower case).

## Donor model level

On this level of the interface module, the data from the donor models is read, interpolated to the exchange grid. The exchange grid is built from a combination of the CRS\_reference coordinates for SeisSol/Shaltop or the lower left corner data for BingClaw, the coordinate data of the donor models and the specified resolution.

Within the interface module, the donor model level is set up as follows: a script from the donorModels directory gets called (get\_donorModel) which in turn calls the respective routine for the donor models - get\_bingclaw for BingClaw, get\_seissol for SeisSol or get\_shaltop for Shaltop - which depends on the donor\_model case. Within each of those routines, the data is read and adapted as written above.

## Exchange grid level

On the exchange grid level, manipulations of the exchange grid data take place.   
  
Within the interface module, scripts from the exchangeGrid directory are called. This includes the filtering functionalities, as well as an interpolation of the provided bathymetry data (given by bathy\_file) to the exchange grid. In this case, we assume that the provided bathymetry data is uniformly stored in a netCDF file.   
The filtering functionalities can either use a Kajiura filter (as implemented for SeisSol/SAMPLER; presented in Maximilian Schmeller’s [bachelor thesis](https://mediatum.ub.tum.de/doc/1554937/1554937.pdf)) or no filter.

## Receiver model level

On the receiver model level, the exchange grid data is written to (new) netCDF files.  
  
Within the interface module, scripts from the receiverModel directory are called. The exchange grid deformation is written using write\_deformation, which in turn uses write2hysea. As of now, only HySEA is a valid receiver.  
Additionally, the interpolated bathymetry (exchange grid) is written as a netCDF file within write\_interpolatedBathy.

# Acknowledgements and contributions

This interface module is a collaborative work by multiple parties. The conceptual idea was developed by Jörn Behrens (UHH). Initial versions of the donor functionalities were provided by Thomas Ulrich (LMU) for SeisSol, Alexis Marboeuf (IPGP) for Shaltop and implemented into the complete interface module by M. Bänsch (UHH). Thomas Ulrich (LMU) provided the filtering resources for the Kajiura filter. Alex González (UMA) provided the knowledge on how to correctly implement the receiver functionalities for HySEA. Valentina Magni (NGI) tested parts of the interface module and gave feedback for additional functionalities.