

EPA SWMM Python adapter

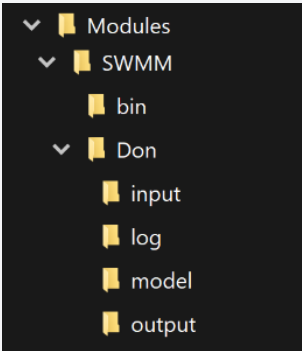
In 2020 a Model Adapter for the EPA SWMM model was developed by Matrix Solutions requested by Toronto and Region Conservation Authority. The Model Adapter was developed in Python 3.8 based on current best practices and the source is published on Github under the MIT license.

The Model Adapter provides the following options and restraints:

- Hydrometeorological forcing is limited to rainfall
- Since EPA SWMM in general is applied as event model, the model adapter does not provide state management
- Conversion of rating curves
- Conversion of time-dependent rule curves
- Conversion of model log messages, in addition to logging of the model adapter
- Conversion of model output (discharges and water levels) to NetCDF:
 - Links: Date, Time, Flow, Velocity, Depth m, Capacity/Setting
 - Nodes: Date, Time, Inflow, Flooding CMS, Depth, Head
- Optional: running of the model

This page describes the configuration of the General Adapter. This example was set-up as a template, where the name of the specific model is inserted via the \$BASIN\$ variable. Information on the functionality of the model adapter, and guidance for preparing the model are described in XXX.

Folder structure

 <pre>graph TD Modules --> SWMM SWMM --> bin SWMM --> Don Don --> input Don --> log Don --> model Don --> output</pre>	<p>The Model Adapter was developed to deal with a model structure as shown here. Both the model adapter and the SWMM executable reside in the bin folder, which is on the same level as model folders. In this example there is one model folder: Don. The model folder contains separate folders for input, logs, model files and output.</p>
---	--

bin: common executable files to all models

- EPA SWMM executable (SWMM5.exe and SWMM5.dll; version 5.1.012)
- Compiled model adapter executable (epaswmm.exe)

<model>: all files relating to a specific EPA SWMM model (i.e. Don River Model).

- Run Information (run_info.xml) in FEWS XML format
- **dump:** zipped folders of the model folder when the model run fails
- **input:** model inputs, exported by FEWS:
 - Rainfall Timeseries (e.g. [rain.nc](#)) in FEWS NetCDF Format
 - Rating Curves (e.g. dam_rating_curve.xml) in FEWS XML format (optional)
 - Control Rules (e.g. control_rules.xml) in FEWS XML format (optional)
- **inputStates:** placeholder directory for potential handling of hotstarts
- **log:**
 - Model adapter logs:
 - Pre-adapter (pre_adapter.log)
 - Post-adapter (post_adapter.log)
 - Model run (run_adapter.log)
 - Output diagnostic file to be imported by FEWS (e.g. diag.xml)
- **model:** EPA SWMM model files
 - Model input file (e.g. DonRiver.inp)
 - Model rainfall data file (e.g. rain.dat)
 - Model output file (e.g. DonRiver.rpt)
 - Model run batch file (run_model.bat)
 - Unit conversion look-up (UDUNITS_lookup.csv)

output: output from the model adapter (converted output for FEWS)

General section

The code block below contains an example for a general section of the General Adapter.

```

<general>
  <description>PCSWM Model for $BASIN$</description>
  <piVersion>1.5</piVersion>
  <!--Root dir set to model folder, because SWMM.exe requires rootDir as currentdir-->
  <rootDir>%REGION_HOME%/Modules/SWMM/$BASIN$</rootDir>
  <workDir>%REGION_HOME%/Modules/SWMM/$BASIN$/model</workDir>
  <exportDir>%ROOT_DIR%/input</exportDir>
  <exportDataSetDir>%ROOT_DIR$</exportDataSetDir>
  <exportIdMap>IdPCSWMM</exportIdMap>
  <importDir>%ROOT_DIR%/output</importDir>
  <importIdMap>IdPCSWMM</importIdMap>
  <importUnitConversionsId>ImportUnitConversions</importUnitConversionsId>
  <dumpFileDir>%GA_DUMPFILEDIR$</dumpFileDir>
  <dumpDir>%ROOT_DIR$</dumpDir>
  <diagnosticFile>%ROOT_DIR%/log/diag.xml</diagnosticFile>
  <missVal>NaN</missVal>
  <timeZone>
    <timeZoneName>GMT</timeZoneName>
  </timeZone>
  <startDateTimeFormat>yyyy-MM-dd hh:mm:ss</startDateTimeFormat>
</general>

```

Some specific settings are required for this model:

- The Rootdir needs to be the model folder. This is essential due to the SWMM executable needing this folder as current dir.
- the WorkDir needs to point to the folder with the model files
- the exportDir and importDir need to point to the input and output folder
- the diagnosticsfile needs to point to the log/diag.xml file
- TimeZone needs to be in GMT, because the Model Adapter expects and writes in GMT time

StartupActivities (optional)

the following settings are optional, but good practice:

```

<startUpActivities>
  <purgeActivity>
    <filter>%ROOT_DIR%/output/*.*</filter>
  </purgeActivity>
  <purgeActivity>
    <filter>%ROOT_DIR%/input/*.*</filter>
  </purgeActivity>
  <purgeActivity>
    <filter>%ROOT_DIR%/log/*.*</filter>
  </purgeActivity>
  <purgeActivity>
    <filter>%ROOT_DIR%/model/*.*</filter>
  </purgeActivity>
</startUpActivities>

```

ExportActivities

Runinfo.xml

The Model Adapter uses the runinfo.xml file to determine the specific features of the model simulation. The Runinfo.xml file needs to be exported to the rootDir.

```

<exportRunFileActivity>
  <exportFile>%ROOT_DIR%/run_info.xml</exportFile>
  <properties>
    <string key="model-executable" value="%REGION_HOME%/Modules/SWMM/bin/swmm5.exe"/>
    <string key="swmm_input_file" value="%WORK_DIR%/BASIN$.inp"/>
  </properties>
</exportRunFileActivity>

```

Next to information such as model folder, start and end time of the model simulation, the Model Adapter needs to additional properties:

- model-executable: path to the SWMM executable, in case the model adapter is used to run the model
- swmm_input_file: the full path to the model file

Rainfall

The following code block contains an example for the export of the rainfall forcing. The export should be a NetCDF file. The Model Adapter will read the filename from the Runinfo.xml:

```
<inputNetcdfFile>C...\input\rain.nc</inputNetcdfFile>
```

```
<exportNetcdfActivity>
  <exportFile>rain.nc</exportFile>
  <timeSeriesSets>
    <timeSeriesSet>
      <moduleInstanceId>$BASIN$_Forecast_preprocess_$NWP$</moduleInstanceId>
      <valueType>scalar</valueType>
      <parameterId>$PARAMETER$</parameterId>
      <locationSetId>$BASIN$_subcatchments</locationSetId>
      <timeSeriesType>simulated forecasting</timeSeriesType>
      <timeStep unit="hour"/>
      <readWriteMode>read complete forecast</readWriteMode>
    </timeSeriesSet>
  </timeSeriesSets>
</exportNetcdfActivity>
```

IdMapping is required to map the average rainfall per subcatchment to the raingauges configured in the model:

```
<function internalLocationSet="Don_subcatchments" externalLocationFunction="Don_@EXTERNAL_ID@"
externalParameterFunction="P" internalParameter="PC.nwp" internalQualifier="DPS"/>
```

Control Rules (optional)

Control rules (time-dependent) can be exported, in order to have some control on structures in the model. The model needs to be set-up specifically to be able to use this functionality. This functionality has been developed aiming at changing the settings of a structure at specific moments: therefore the exported timeseries can be non-equidistant (equidistant is also possible).

```
<exportTimeSeriesActivity>
  <exportFile>Control_rules.xml</exportFile>
  <timeSeriesSets>
    <timeSeriesSet>
      <moduleInstanceId>Don_Forecast</moduleInstanceId>
      <valueType>scalar</valueType>
      <parameterId>Fraction</parameterId>
      <locationId>HY027w1</locationId>
      <timeSeriesType>external historical</timeSeriesType>
      <timeStep unit="nonequidistant"/>
      <relativeViewPeriod unit="day" start="-1" end="2" startOVERRULABLE="false"
endOVERRULABLE="true"/>
      <readWriteMode>add originals</readWriteMode>
    </timeSeriesSet>
  </timeSeriesSets>
</exportTimeSeriesActivity>
```

The file will only be read by the Model Adapter if it is named **Control_rules.xml**.

An Id map is required to map the location and structure type with that in the model. The external parameter needs to be the object type.

```
<map internalLocation="HY027w1" internalParameter="Fraction" externalLocation="OL341" externalParameter="OUTLET" />
```

the following object types are supported: pump, orifice, weir or outlet.

Rating Curves (optional)

Rating curves can be exported to replace existing rating curves in the model.

```
<exportRatingCurveActivity>
  <exportFile>Dam_rating_curve.xml</exportFile>
  <locationId>HY027w1</locationId>
</exportRatingCurveActivity>
```

The model adapter will check the runinfo.xml file for an <inputRatingCurveFile>element. If so, it will try to read and convert the content of the file.

Notice that an IdMapping is necessary to map the internal location with the Id in the model:

```
<location internal="HY027w1" external="GRossDam" />
```

Export ModuleDataSet (optional)

It is possible to export a ModuleDataSet containing the model file itself, directly into the model folder. This can be used to have the forecaster a model file with a different initial model state (eg dry, normal, wet). These files need to be included in the ModuleDataSets folder in the config folder, and can be configured as [what-if scenarios](#).

```
<exportDataSetActivity>
  <moduleInstanceId>$moduledataset$</moduleInstanceId>
</exportDataSetActivity>
```

Notice that the Model Adapter is not involved in this activity.

Execute Activities

The General Adapter has three execute activities:

1. run the pre adapter
2. run the model
3. run the post adapter

Pre adapter

First the Model Adapter is started. It gets the location of the runinfo.xml passed as argument, as well as "pre" indicating it should run as preadapter.

```
<executeActivity>
  <command>
    <executable>%REGION_HOME%/Modules/SWMM/bin/epaswmm.exe</executable>
  </command>
  <arguments>
    <argument>--run_info</argument>
    <argument>%ROOT_DIR%/run_info.xml</argument>
    <argument>pre</argument>
  </arguments>
  <timeOut>5000000</timeOut>
</executeActivity>
```

Model simulation

The next activity is starting the model simulation itself. This can be done by the model adapter itself, but to keep control it is also possible to run the SWMM executable independently. The SWMM exe needs two arguments: the model file itself, and the name of the file in which the simulation output need to be written.

```
<executeActivity>
  <command>
    <executable>%REGION_HOME%/Modules/SWMM/bin/SWMM5.exe</executable>
  </command>
  <arguments>
    <argument>%WORK_DIR%/$BASIN$.inp</argument>
    <argument>%WORK_DIR%/$BASIN$.rpt</argument>
  </arguments>
  <timeOut>5000000</timeOut>
  <ignoreDiagnostics>true</ignoreDiagnostics>
</executeActivity>
```

Notice that ignoreDiagnostics needs to be set to true, since the model does not return a log in PI-XML format.

Post Adapter

After the model simulation the Model Adapter will be executed again, this time with argument "post". It will convert the model output to NetCDF format and the model log messages to PI-XML.

```
<executeActivity>
  <command>
    <executable>%REGION_HOME%/Modules/SWMM/bin/epaswmm.exe</executable>
  </command>
  <arguments>
    <argument>--run_info</argument>
    <argument>%ROOT_DIR%/run_info.xml</argument>
    <argument>post</argument>
  </arguments>
  <timeOut>5000000</timeOut>
</executeActivity>
```

Import model results

The ImportActivities only include the reading of the output timeseries. Notice that the importFiles have the name of the model file with postfix "output_nodes" and "output_links". The Model Adapter has converted all the output of the model: with ID mapping in FEWS can be determined what model results are actually imported.

```

<importPiNetcdfActivity>
  <importFile>$BASIN$_output_nodes.nc</importFile>
  <timeSeriesSets>
    <timeSeriesSet>
      <moduleInstanceId>$BASIN$_Forecast_$NWP$</moduleInstanceId>
      <valueType>scalar</valueType>
      <parameterId>H.sim</parameterId>
      <qualifierId>$NWP$</qualifierId>
      <locationSetId>SWMM_$BASIN$_H</locationSetId>
      <timeSeriesType>simulated forecasting</timeSeriesType>
      <timeStep unit="minute" multiplier="15"/>
      <readWriteMode>add originals</readWriteMode>
    </timeSeriesSet>
  </timeSeriesSets>
</importPiNetcdfActivity>
<importPiNetcdfActivity>
  <importFile>$BASIN$_output_links.nc</importFile>
  <timeSeriesSets>
    <timeSeriesSet>
      <moduleInstanceId>$BASIN$_Forecast_$NWP$</moduleInstanceId>
      <valueType>scalar</valueType>
      <parameterId>Q.sim</parameterId>
      <qualifierId>$NWP$</qualifierId>
      <locationSetId>SWMM_$BASIN$_Q</locationSetId>
      <timeSeriesType>simulated forecasting</timeSeriesType>
      <timeStep unit="minute" multiplier="15"/>
      <readWriteMode>add originals</readWriteMode>
    </timeSeriesSet>
  </timeSeriesSets>
</importPiNetcdfActivity>
</importActivities>

```

Of course also here IDmapping is required. Important to notice is that SWMM exports the results at nodes and links with a prefix "Node_", "Link_" respectively, what is also reflected in the example below.

```

<function internalLocationSet="SWMM_Don_H" externalLocationFunction="Node_@ID_SWMM_H@"
externalParameterFunction="Head" internalParameter="H.sim"/>
<function internalLocationSet="SWMM_Don_Q" externalLocationFunction="Link_@ID_SWMM_Q@"
externalParameterFunction="Flow" internalParameter="Q.sim"/>

```