OpenM++ wiki

Table of contents

Home

Getting Started

- **Windows**: Quick Start for Model Users
- **Windows**: Quick Start for Model Developers
- **Linux**: Quick Start for Model Users
- **Linux**: Quick Start for Model Developers
- **MacOS**: Quick Start for Model Users
- **MacOS**: Quick Start for Model Developers
- Model Run: How to Run the Model
- MIT License, Copyright and Contribution

Model development in OpenM++

- Model Code: Programming a model
- **Windows**: Create and Debug Models
- **Linux**: Create and Debug Models
- **MacOS**: Create and Debug Models
- **MacOS**: Create and Debug Models using Xcode
- **Modgen**: Convert case-based model to openM++
- **Modgen**: Convert time-based model to openM++
- **Modgen**: Convert Modgen models and usage of C++ in openM++ code
- Model Localization: Translation of model messages

Using OpenM++

- How To: Set Model Parameters and Get Results
- Model Run: How model finds input parameters
- Model Output Expressions
- Model Run Options and ini-file
- OpenM++ Compiler (omc) Run Options
- OpenM++ ini-file format
- UI: How to start user interface
- UI: openM++ user interface
- UI: Create new or edit scenario
- UI: Upload input scenario or parameters
- UI: Run the Model
- UI Localization: Translation of openM++

OpenM++ web-service: API and cloud setup

- Oms: openM++ web-service
- Oms: openM++ web-service API
- Oms: How to prepare model input parameters
- Oms: Cloud and model runs queue

Using OpenM++ from Python and R

- Use R to save output table into CSV file
- Use R to save output table into Excel
- Run model from Python: simple loop over model parameter

Run RiskPaths model from Python: advanced parameters scaling

Run model from R: simple loop over model parameter

Run model from R: simple loop in cloud

Run RiskPaths model from R: advanced parameters scaling

Run RiskPaths model from R: advanced run in cloud

Run model from R and save results in CSV file

Docker

Windows: Use Docker to get latest version of OpenM++

Linux: Use Docker to get latest version of OpenM++

RedHat 8: Use Docker to get latest version of OpenM++

OpenM++ Development

Quick Start for OpenM++ Developers

Setup Development Environment

2018, June: OpenM++ HPC cluster: Test Lab

Development Notes: Defines, UTF-8, Databases, etc.

OpenM++ Design, Roadmap and Status

2012, December: OpenM++ Design

2012, December: OpenM++ Model Architecture, December 2012

2012, December: Roadmap, Phase 1

2013, May: Prototype version

2013, September: Alpha version

2014, March: Project Status, Phase 1 completed

2016, December: Task List

2017, January: Design Notes. Subsample As Parameter problem. Completed

OpenM++ web-service API

Oms: openM++ web-service

Oms: openM++ web-service API

GET Model Metadata

GET model list

GET model list including text (description and notes)

GET model definition metadata

GET model metadata including text (description and notes)

GET model metadata including text in all languages

GET Model Extras

GET model languages

GET model language-specific strings

GET model profile

GET list of profiles

GET Model Run results metadata

GET list of model runs

GET list of model runs including text (description and notes)

GET status of model run

GET status of model run list

GET status of first model run

GET status of last model run

GET status of last completed model run

GET model run metadata and status

GET model run including text (description and notes)

GET model run including text in all languages

GET Model Workset metadata: set of input parameters

GET list of model worksets

GET list of model worksets including text (description and notes)

GET workset status

GET model default workset status

GET workset including text (description and notes)

GET workset including text in all languages

Read Parameters, Output Tables or Microdata values

Read parameter values from workset

Read parameter values from workset (enum id's)

Read parameter values from model run

Read parameter values from model run (enum id's)

Read output table values from model run

Read output table values from model run (enum id's)

Read microdata values from model run

Read microdata values from model run (enum id's)

GET Parameters, Output Tables or Microdata values

GET parameter values from workset

GET parameter values from model run

GET output table expression(s) from model run

GET output table accumulator(s) from model run

GET output table all accumulators from model run

GET microdata values from model run

GET Parameters, Output Tables or Microdata as CSV

GET csv parameter values from workset

GET csv parameter values from workset (enum id's)

GET csv parameter values from model run

GET csv parameter values from model run (enum id's)

GET csv output table expressions from model run

GET csv output table expressions from model run (enum id's)

GET csv output table accumulators from model run

GET csv output table accumulators from model run (enum id's)

GET csv output table all accumulators from model run

GET csv output table all accumulators from model run (enum id's)

GET csv microdata values from model run

GET csv microdata values from model run (enum id's)

GET Modeling Task metadata and task run history

GET list of modeling tasks
GET list of modeling tasks including text (description and notes)
GET modeling task input worksets
GET modeling task run history
GET status of modeling task run
GET status of modeling task run list
GET status of modeling task first run
GET status of modeling task last run
GET status of modeling task last completed run
GET modeling task including text (description and notes)
GET modeling task text in all languages

Update Model Profile: set of key-value options

PATCH create or replace profile
DELETE profile
POST create or replace profile option
DELETE profile option

Update Model Workset: set of input parameters

POST update workset read-only status
PUT create new workset
PUT create or replace workset
PATCH create or merge workset
DELETE workset
DELETE parameter from workset
PATCH update workset parameter values
PATCH update workset parameter values (enum id's)
PATCH update workset parameter(s) value notes
PUT copy parameter from model run into workset
PATCH merge parameter from model run into workset
PUT copy parameter from workset to another
PATCH merge parameter from workset to another

Update Model Runs

PATCH update model run text (description and notes)
DELETE model run
PATCH update run parameter(s) value notes

Update Modeling Tasks

PUT create or replace modeling task
PATCH create or update modeling task
DELETE modeling task

Run Models: run models and monitor progress

POST a request to run the model

GET state of current model run

PUT stop model run

Download model, model run results or input parameters

GET download log file

GET model download log files

GET all download log files

GET download files tree

POST initiate entire model download

POST initiate model run download

POST initiate model workset download

DELETE download files

DELETE download files asynchronously

Upload model runs or worksets (input scenarios)

GET upload log file

GET all upload log files for the model

GET all upload log files

GET upload files tree

POST initiate model run upload

POST initiate workset upload

DELETE upload files

DELETE upload files asynchronously

User: manage user settings and data

GET user views for the model

PUT user views for the model

DELETE user views for the model

Model run jobs and service state

GET web-service configuration

GET web-service state

GET state of active model run job

GET state of model run job from queue

GET state of model run job from history

PUT model run job into other queue position

DELETE state of model run job from history

Administrative: manage web-service state

POST a request to refresh models catalog

POST a request to close models catalog

POST a request to pause model run queue

PUT a request to shutdown web-service

Home

This is the home of the OpenM++ wiki. It consists mostly of links to other topics, organized into sections. For a brief description of what OpenM++ can bring to a micro-simulation or agent-based modelling project please see the Features section. Our Glossary contains brief explanations of some of the terms used in this wiki.

Quick links

- Table of Contents
- Glossary
- Contact us

Contents

- Introduction to OpenM++
- Getting started
- Model development
- Model use
- Model API
- Model scripting
- Cloud or cluster installation and Docker
- Features
- *for programmers:* OpenM++ development
- *for programmers:* OpenM++ design
- *for programmers:* OpenM++ source code
- Contact us

Introduction to OpenM++

OpenM++ is an open source platform to develop, use, and deploy micro-simulation or agent-based models. OpenM++ was designed to enable non-programmers to develop simple or complex models. Click here for an overview of OpenM++ features.

[back to contents]

Getting started

This section describes how to get OpenM++ installed and working on Windows, Linux, or MacOS, for model users or for model developers. The installation kits include a collection of simple illustrative models. That same collection of models is also present in the cloud, where it can be accessed from any web browser, with no installation required. For more information on the OpenM++ cloud collection, please Contact us.

- Download OpenM++ for Windows, Linux or MacOS↗
- **Windows**: Quick Start for Model Users
- **Windows**: Quick Start for Model Developers
- **Linux**: Quick Start for Model Users
- **Linux**: Quick Start for Model Developers
- **MacOS**: Quick Start for Model Users
- **MacOS**: Quick Start for Model Developers
- Model Run: How to Run the Model

[back to contents]

Model development

Platform-independent information:

- Model Development Topics: A list of topics related to model development in OpenM++

Platform-specific information:

- **Windows**: Create and Debug Models
- **Linux**: Create and Debug Models
- **MacOS**: Create and Debug Models
- **MacOS**: Create and Debug Models using Xcode

Modgen-specific information:

- **Modgen**: Convert case-based model to openM++
- **Modgen**: Convert time-based model to openM++
- **Modgen**: Convert Modgen models and usage of C++ in openM++ code

[back to contents]

Model use

This section describes how to use a model once built.

- How To: Set Model Parameters and Get Results
- Model Data Import-Export: How to Use dbcopy↗
- Model Run: How model finds input parameters
- Model Output Expressions
- Model Run Options and ini-file
- OpenM++ ini-file format
- UI: How to start user interface
- UI: openM++ user interface
- UI: Create new or edit scenario
- UI: Upload input scenario or parameters
- UI: Run the Model
- UI Localization: Translation of openM++

Modgen-specific information:

- **Modgen**: CsvToDat utility: Command-line utility to convert CSV parameters to DAT format

[back to contents]

Model API and run models in cloud

The model API provides programmatic access to scenario management, model inputs, model runs, and model outputs. It is implemented by the OpenM++ `oms` web service, which uses standard JSON to communicate with a controlling application. The worked examples in Model scripting provide practical illustrations of how to use the model API and the `oms` service to automate an analysis. Incidentally, the browser-based OpenM++ user interface uses the model API and the `oms` service for all model-specific operations. It is also possible to create workspace for model users in cloud using `oms` web-service.

- Oms: openM++ web-service
- Oms: openM++ web-service API
- Oms: How to prepare model input parameters
- Oms: Cloud and model runs queue
- Documentation and source code: Go library and tools↗

[back to contents]

Model scripting

The topics in this section illustrate model-based analysis in two different scripting environments: Python and R. The Model API is used in these environments to create scenarios, run the model iteratively, and retrieve results for graphical presentation in the scripting environment.

- Use R to save output table into CSV file
- Use R to save output table into Excel
- Run model from Python: simple loop over model parameter
- Run RiskPaths model from Python: advanced parameters scaling
- Run model from R: simple loop over model parameter
- Run model from R: simple loop in cloud
- Run RiskPaths model from R: advanced parameters scaling
- Run RiskPaths model from R: advanced run in cloud
- Run model from R and save results in CSV file
- OpenMpp R package documentation↗

[back to contents]

Docker

Docker is a technology used here to quickly replicate preconfigured operating system environments containing OpenM++ functionality.

- Windows: Use Docker to get latest version of OpenM++
- Linux: Use Docker to get latest version of OpenM++
- RedHat 8: Use Docker to get latest version of OpenM++
- DockerHub: image to run openM++ models↗
- DockerHub: image to build latest openM++ version↗

[back to contents]

Features

Here is a summary of some OpenM++ features:

General features:

- open source: OpenM++ and all components are licensed under the very broad MIT license.
- cross-platform: Model development and use on Windows, Linux, or MacOS.
- standards-based: Uses industry standard formats and technologies.
- zero-footprint: File-based installation requires no elevation of privileges.

Model developer features:

- high-level language: Model types, parameters, entities, events, tables, etc. are specified using a compact domain-specific language targeted to microsimulation.
- scalable complexity: From simple 'toy' models to highly complex models.
- modularity: New events and processes can be added to a model in a new module, often with little or no modification to existing modules.
- continuous or discrete time, or a mixture.
- supports multiple versions: Multiple OpenM++ versions can be installed and a single environment variable used to choose among them.
- result compare: Supports rapid comparison of all model outputs during incremental model development.

Computational features:

- scalable computation: Designed to scale linearly with population size or replicates when possible, N log N scaling for typical interacting populations.
- grid-enabled, cloud-enabled: Supports MPI for multi-processing to distribute execution of replicates to a small or large computational grid or to the cloud, with automatic result assembly.
- multi-threaded: Supports multi-threading for parallel execution of replicates on desktop or server.
- on-the-fly tabulation: Tables are computed during the simulation, eliminating the need to output voluminous microdata for subsequent tabulation.
- computationally efficient: The model specification is transformed to C++ which is processed by an optimizing C++ compiler to produce a highly efficient executable program.

Usability features:

- generated UI: A model-specific UI is generated from the model specification.
- browser-based UI: The UI requires only a browser, and runs on almost any modern browser.
- cloud-enabled: Models can be deployed to a cloud and accessed remotely over the web, from a browser.
- multilingual support: For UI and for model, with real-time language switching

Analyst features:

- continuous time tabulation: Powerful but easy to use language constructs to tabulate time-in-state, empirical hazards, transitions counts, state changes, etc.
- replicate support: All tables can have underlying replicate simulations to assess the uncertainty of any cell of any output table. Statistical measures of uncertainty are computed for all cells of all tables.
- automation: Models can be controlled by scripts, eg Python or R.
- import/export: Models and runs can be moved between databases, or to standard formats for upstream preparation of inputs or for downstream analysis of outputs.
- dynamic run control: A computational grid can process runs dynamically to enable whole-model estimation or calibration, with a controlling script reading run results and preparing new runs for execution.

The OpenM++ language is based on the Modgen↗ language developed at Statistics Canada. With minor modifications to model source code, existing Modgen models can work with either Modgen or OpenM++.

[back to contents]

OpenM++ development

This section contains technical information for programmers interested in OpenM++ itself, as opposed to model developers or model users. It describes how to set up a programming environment to build and modify OpenM++.

- Quick Start for OpenM++ Developers
- Setup Development Environment

- 2018, June: OpenM++ HPC cluster: Test Lab
- Development Notes: Defines, UTF-8, Databases, etc.

[back to contents]

OpenM++ design

This section contains technical and project information of interest to programmers or system architects. It dates from the inception and 'alpha' days of the OpenM++ project. The road map diagram remains somewhat relevant and may be useful for a broad overview of the major components of OpenM++ from the perspective of a programmer or system architect.

Project Status: production stable since February 2016

- 2012, December: OpenM++ Design
- 2012, December: OpenM++ Model Architecture, December 2012
- 2012, December: Roadmap, Phase 1
- 2013, May: Prototype version
- 2013, September: Alpha version
- 2014, March: Project Status, Phase 1 completed
- 2016, December: Task List
- 2017, January: Design Notes. Subsample As Parameter problem. Completed

[back to contents]

OpenM++ source code

This section contains technical information for programmers interested in OpenM++ itself, as opposed to model developers or model users. It contains links to the OpenM++ source code and to the documentation of that source code.

- GitHub: Run-time and compiler c++ Source code↗
- Source code documentation: Runtime library↗
- Source code documentation: Compiler↗
- GitHub: Go library, web-service and db tools Source Code↗
- Source code documentation: Go library and tools↗
- GitHub: openMpp R package↗
- Source code documentation: openMpp R package↗
- GitHub: Source code to build Docker images↗
- GitHub: OpenM++ UI frontend↗

[back to contents]

Contact Us

- OpenM++ web-site↗
- E-mail: `openmpp dot org at gmail dot com`
- License, Copyright and Contribution: OpenM++ is Open Source and Free
- MIT License↗
- OpenM++ on GitHub↗
- OpenM++ on DockerHub↗

[back to contents]

Windows: Quick Start for Model Users

Where is OpenM++

- Download:
    - desktop version: binary files and source code openmpp_win_YYYYMMDD.zip
    - cluster version: binary files and source code openmpp_win_mpi_YYYYMMDD.zip
    - Docker image to run openM++ models: openmpp/openmpp-run
- Documentation: this wiki

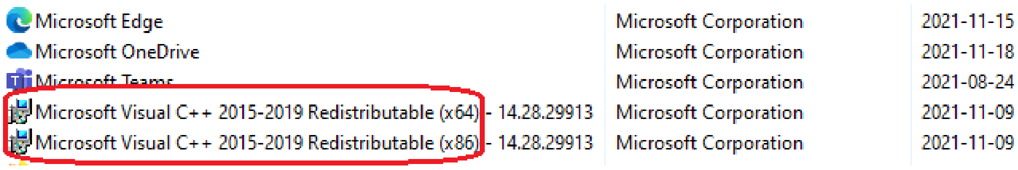
It is recommended to start from desktop version of openM++.

You need to use cluster version of openM++ to run the model on multiple computers in your network, in cloud or HPC cluster environment. OpenM++ is using MPI to run the models on multiple computers. Please check Model Run: How to Run the Model page for more details.

You can use Docker containers to avoid installation of multiple additional components in your host computer. Because all necessary software will be installed in container your host system will be clean.

Prerequisites

You may need to install Microsoft Visual C++ redistributable runtime, unless it is already installed as a part of some other software package:



If it is not present then please follow Microsoft instructions about: Visual C++ Redistributable.

Run on Windows computer

- download and unzip Windows desktop binaries openmpp_win_YYYYMMDD.zip into C:\SomeDir\
- run modelOne model with single subsample on local machine:

```
C:
cd \SomeDir\openmpp_win_20180205\models\bin
modelOne.exe
```

```
2014-03-17 17:14:24.0023 Model: modelOne
2014-03-17 17:14:24.0070 Reading Parameters
2014-03-17 17:14:24.0085 Running Simulation
2014-03-17 17:14:24.0101 Writing Output Tables
2014-03-17 17:14:24.0179 Done.
```

- run modelOne model with 16 subsamples and 4 threads:

```
modelOne.exe -OpenM.Subvalues 16 -OpenM.Threads 4
```

```
2017-06-06 17:35:29.0421 modelOne
2017-06-06 17:35:29.0435 One-time initialization
2017-06-06 17:35:29.0454 Run: 106
2017-06-06 17:35:29.0456 Reading Parameters
2017-06-06 17:35:29.0460 Running Simulation
2017-06-06 17:35:29.0464 Writing Output Tables
........................
2017-06-06 17:35:29.0870 Done.
```

- run other models (i.e. NewCaseBased, NewTimeBased, RiskPaths):

```
NewCaseBased.exe -OpenM.Subvalues 8 -OpenM.Threads 2
```

- run RiskPaths model with new parameter value `CanDie = true` and all other parameter values the same as in previous model run:

```
RiskPaths.exe -Parameter.CanDie true -OpenM.BaseRunId 102
```

```
2020-08-14 17:27:48.574 RiskPaths
2020-08-14 17:27:48.610 Run: 103
2020-08-14 17:27:48.618 Sub-value: 0
2020-08-14 17:27:48.628 member=0 Simulation progress=0% cases=0
........................
2020-08-14 17:27:54.883 Done.
```

- run modelOne to compute modeling task "taskOne":

  ```
  modelOne.exe -OpenM.Subvalues 16 -OpenM.Threads 4 -OpenM.TaskName taskOne
  ```

  ```
  2017-06-06 17:39:24.0757 modelOne
  2017-06-06 17:39:24.0782 One-time initialization
  2017-06-06 17:39:24.0800 Run: 107
  2017-06-06 17:39:24.0802 Reading Parameters
  2017-06-06 17:39:24.0807 Running Simulation
  ........................
  2017-06-06 17:39:25.0232 Run: 108
  2017-06-06 17:39:25.0234 Reading Parameters
  ........................
  2017-06-06 17:39:25.0661 Done.
  ```

- in case if previous model run fail, for example, due to power outage, then it can be "restarted":

  ```
  modelOne.exe -OpenM.RestartRunId 1234
  ```

  output may vary depending on the stage where previous modelOne run failed, but still similar to above.

Note: We recommend to use normal Windows command line cmd.exe. If you are using Windows PowerShell then it may be necessary to put "quotes" around command line options, e.g:

```
model.exe "-OpenM.Subvalues" 16
```

Run on multiple computers over network, in HPC cluster or cloud

- download and unzip Windows cluster binaries openmpp_win_mpi_YYYYMMDD.zip into C:\AnyDir. Please notice name of cluster version archive has ***mpi*** in it, i.e. `openmpp_win_mpi_20180205.zip` and is located in a subdirectory **mpi**.
- if you are using regular Windows computers in your organization network (like Windows 7 or 10 and not MS HPC servers or Azure) then:
  - make sure you have latest version of Microsoft MPI Redistributable installed.
  - or pull Docker image `docker pull openmpp/openmpp-run:windows-1903` to run models inside the container (see below).
- run modelOne model with single subsample on local machine:

  ```
  C:
  cd \AnyDir\openmpp_win_mpi_20180205\models\bin
  modelOne_mpi.exe
  ```

  ```
  2014-03-17 17:14:24.0023 Model: modelOne
  2014-03-17 17:14:24.0070 Reading Parameters
  2014-03-17 17:14:24.0085 Running Simulation
  2014-03-17 17:14:24.0101 Writing Output Tables
  2014-03-17 17:14:24.0179 Done.
  ```

- run two instances of modelOne to compute 16 subsamples and 4 threads:

  ```
  mpiexec -n 2 modelOne_mpi.exe -OpenM.Subvalues 16 -OpenM.Threads 4
  ```

```
2017-06-06 17:52:06.0143 modelOne
2017-06-06 17:52:06.0145 modelOne
2017-06-06 17:52:06.0179 Parallel run of 2 modeling processes, 4 thread(s) each
2017-06-06 17:52:06.0179 One-time initialization
2017-06-06 17:52:06.0179 One-time initialization
2017-06-06 17:52:06.0192 Run: 106
2017-06-06 17:52:06.0192 Run: 106
2017-06-06 17:52:06.0192 Reading Parameters
........................
2017-06-06 17:52:06.0532 Writing Output Tables
2017-06-06 17:52:06.0599 Done.
2017-06-06 17:52:06.0599 Done.
```

- run other models (i.e. NewCaseBased, NewTimeBased, RiskPaths):

```
mpiexec -n 8 NewCaseBased_mpi.exe -OpenM.Subvalues 64 -OpenM.Threads 4
```

Microsoft recommends to install HPC Pack which simplifies your computational resources management rather than using `mpiexec` as above. It is also possible to use Microsoft Azure cloud where compute nodes available for you on demand.

Run models using Docker container

- download and unzip openmpp_win_YYYYMMDD.zip into C:\AnyDir.
- make sure you have Docker for Windows installed, see Microsoft documentation for more details.
- pull Docker image:

```
docker pull openmpp/openmpp-run:windows-1903
```

- run modelOne model with single subsample:

```
docker run -v C:\AnyDir\models\bin:C:\ompp openmpp/openmpp-run:windows-1903 modelOne.exe
```

```
2014-03-17 17:14:24.0023 Model: modelOne
2014-03-17 17:14:24.0070 Reading Parameters
2014-03-17 17:14:24.0085 Running Simulation
2014-03-17 17:14:24.0101 Writing Output Tables
2014-03-17 17:14:24.0179 Done.
```

- run two instances of modelOne to compute 16 subsamples and 4 threads:

```
docker run -v C:\AnyDir\models\bin:C:\ompp openmpp/openmpp-run:windows-1903 mpiexec -n 2 modelOne_mpi.exe -OpenM.Subvalues 16 -OpenM.Threads 4
```


```
2017-06-06 17:52:06.0143 modelOne
2017-06-06 17:52:06.0145 modelOne
2017-06-06 17:52:06.0179 Parallel run of 2 modeling processes, 4 thread(s) each
2017-06-06 17:52:06.0179 One-time initialization
2017-06-06 17:52:06.0179 One-time initialization
2017-06-06 17:52:06.0192 Run: 106
2017-06-06 17:52:06.0192 Run: 106
2017-06-06 17:52:06.0192 Reading Parameters
........................
2017-06-06 17:52:06.0532 Writing Output Tables
2017-06-06 17:52:06.0599 Done.
2017-06-06 17:52:06.0599 Done.
```


- run other models (i.e. NewCaseBased, NewTimeBased, RiskPaths):

```
docker run -v C:\AnyDir\models\bin:C:\ompp openmpp/openmpp-run:windows-1903 mpiexec -n 8 NewCaseBased_mpi.exe -OpenM.Subvalues 64 -OpenM.Threads 4
```

Windows: Quick Start for Model Developers

Step by Step

- Download desktop version zip archive: openmpp_win_YYYYMMDD.zip binary files and source code
- Extract zip archive to C:\openmpp_win_20210112\
- Build the example RiskPaths model and run the Default scenario
  - Open C:\openmpp_win_20210112\models\RiskPaths\RiskPaths-ompp.sln using Visual Studio 2019
  - 'Rebuild' in Visual Studio 2019 to build the model and run the Default scenario
  - (optional) Enable in project Properties -> OpenM++ -> "Run scenario after build" to examine model run results
  - (optional) Enable "Export model run results into csv files" to create CSV files containing values of model parameters and output tables
  - (optional) Enable "Open model web UI" to modify parameters, run the model and view model results
- (optional) to enable model development from any directory, independent of `C:\openmpp_win_20210112` location, do any of:
  - open a Command Prompt window and type the command: `setx OM_ROOT C:\openmpp_win_20210112`
  - open your model Model.vcrproj file in Notepad and update line:

```
<OM_ROOT>C:\openmpp_win_20210112</OM_ROOT>
```

- How to: create and debug models on Windows

OpenM++ Models: desktop? clusters? MPI?

It is recommended to start from desktop version of openM++.

You need to use cluster version of openM++ to run the model on multiple computers in your network, in cloud or HPC cluster environment. OpenM++ is using MPI to run the models on multiple computers. Please check Model Run: How to Run the Model page for more details.

Build on Windows

Tested platforms:

- Windows 10, 2016 (64 bit)
- Visual Studio 2019, including Community Edition
- (optional) Microsoft MPI SDK Redistributable Package

Note: It may work on any Windows 7 and above or 2008R2 and above, 32 and 64 bits, with Visual Studio 2017. However we are not testing it on older versions of Windows or Visual Studio.

Build debug version of the model

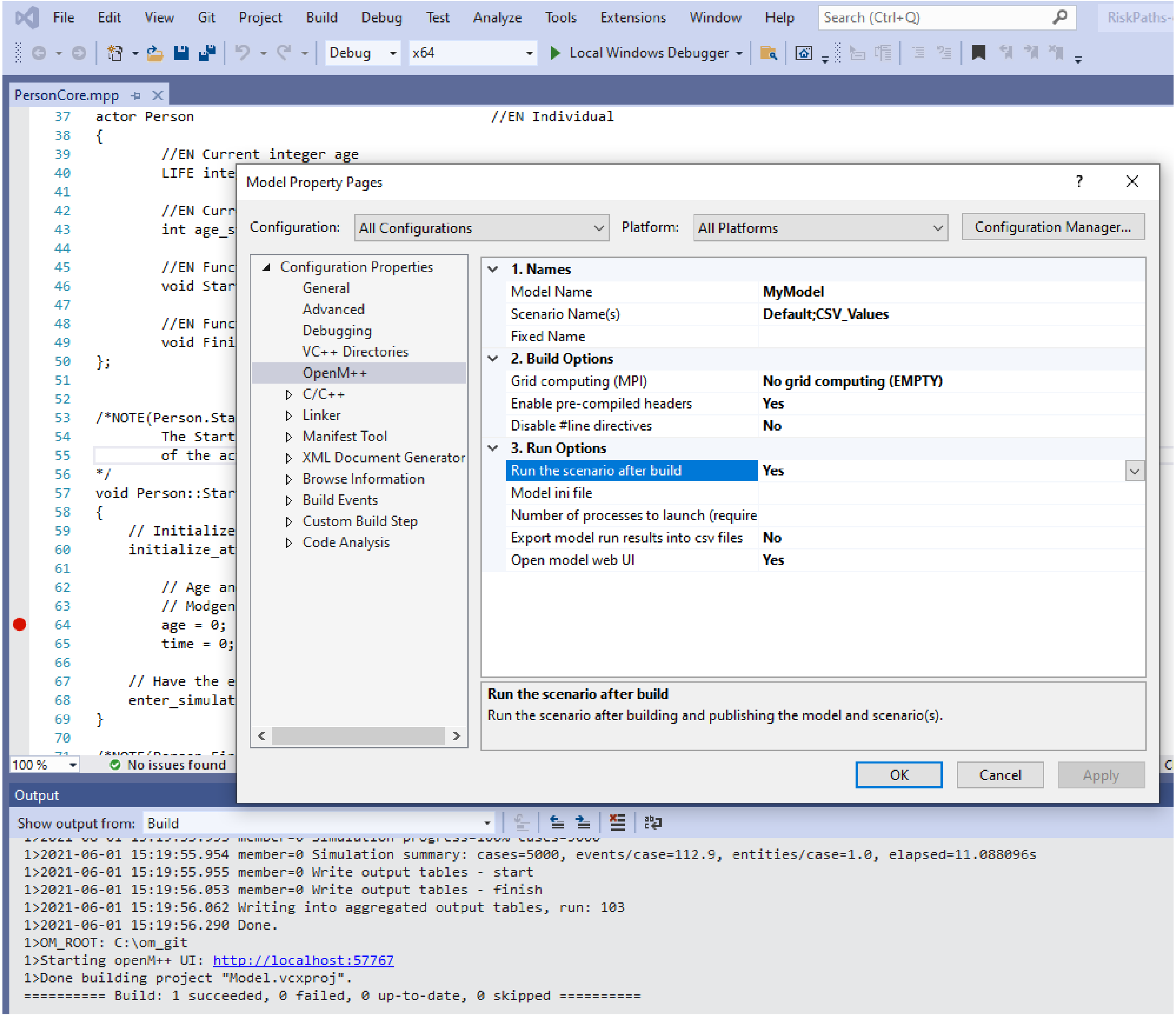
You can use any of test models solution, except of modelOne, as starting point to develop your own model. Below we are using NewCaseBased model as example.

To build and run **debug version** of the model use desktop (non-MPI) version of openM++:

- download and unzip openmpp_win_YYYYMMDD.zip Windows desktop binaries into C:\openmpp_win_20210112\
- build Debug version of the model using solution: `C:\openmpp_win_20210112\models\NewCaseBased\NewCaseBased-ompp.sln`
- (optional) Rebuild the model and run it:
  - go to menu: Project -> Properties -> Configuration Properties -> OpenM++
  - change: Run Options -> Run the scenario after build -> Yes
  - Rebuild project

At bottom Output window of Visual Studio you will see something like:

```
1>Model.vcxproj -> C:\openmpp_win_20210112\models\NewCaseBased\ompp\bin//NewCaseBasedD.exe
1>2017-06-06 18:21:08.0092 NewCaseBased
1>2017-06-06 18:21:08.0160 Run: 102
1>2017-06-06 18:21:08.0163 Get fixed and missing parameters
1>2017-06-06 18:21:08.0166 Get scenario parameters
1>2017-06-06 18:21:08.0172 Sub-value 0
1>2017-06-06 18:21:08.0175 compute derived parameters
1>2017-06-06 18:21:08.0177 Initialize invariant entity data
1>2017-06-06 18:21:08.0180 Member=0 simulation progress=0%
........
1>2017-06-06 18:21:08.0688 member=0 write output tables - finish
1>2017-06-06 18:21:08.0697 Writing Output Tables Expressions
1>2017-06-06 18:21:08.0727 Done.
1>Done building project "Model.vcxproj".
======= Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =======
```



Build cluster version of the model to run on multiple computers over network

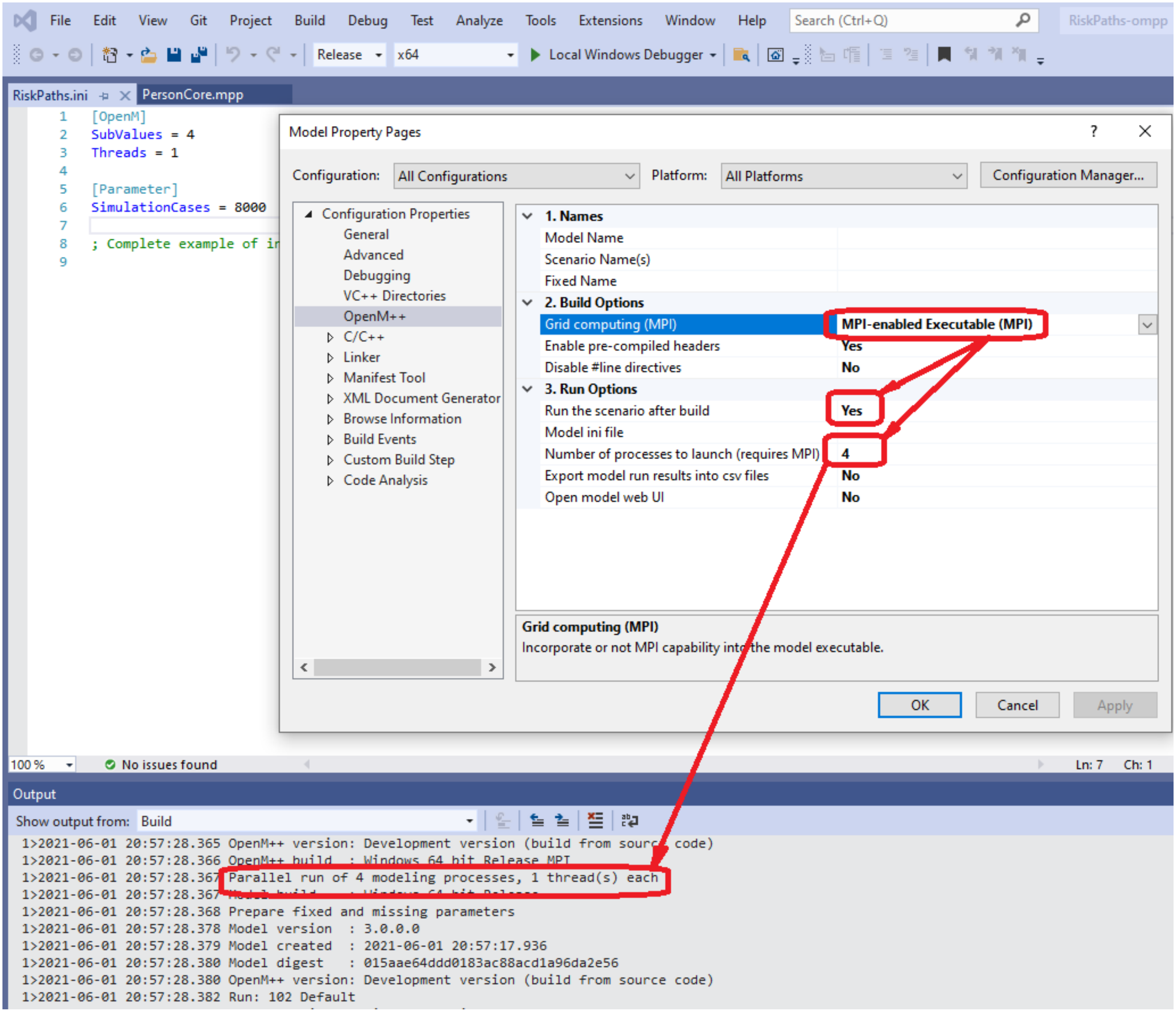
Make sure you have latest version of Microsoft MPI SDK and MPI Redistributable installed.

- download and unzip openmpp_win_mpi_YYYYMMDD.zip Windows cluster binaries into C:\openmpp_win_mpi_20180205. Please notice name of cluster version archive has ***mpi*** in it, i.e. `openmpp_win_mpi_20180205.zip`.
- Rebuild the model and run it:
  - go to menu: Project -> Properties -> Configuration Properties -> OpenM++
  - change: Build Options -> Grid computing (MPI) -> MPI-enabled Executable (MPI)

- change: Run Options -> Number of processes to launch -> *use 2 or more (depends on your cluster configuration)*
- change: Run Options -> Run the scenario after build -> Yes
- Rebuild Model project

At bottom Output window of Visual Studio you will see something like:

```
1>Model.vcxproj -> C:\openmpp_win_mpi_20180205\models\RiskPaths\ompp\bin\RiskPaths_mpi.exe
1>2021-06-01 20:57:28.146 RiskPaths
1>2021-06-01 20:57:28.146 RiskPaths
1>2021-06-01 20:57:28.146 RiskPaths
1>2021-06-01 20:57:28.163 RiskPaths
........
1>2021-06-01 20:57:28.366 OpenM++ build  : Windows 64 bit Release MPI
1>2021-06-01 20:57:28.367 Parallel run of 4 modeling processes, 1 thread(s) each
........
1>2021-06-01 20:57:28.859 member=3 Simulation progress=100% cases=2000
1>2021-06-01 20:57:28.867 member=3 Simulation summary: cases=2000, events/case=112.9, entities/case=1.0, elapsed=0.453989s
1>2021-06-01 20:57:28.868 member=3 Write output tables - start
1>2021-06-01 20:57:28.873 member=3 Write output tables - finish
1>2021-06-01 20:57:29.233 member=0 Write output tables - finish
1>2021-06-01 20:57:29.919 Writing into aggregated output tables, run: 102
1>2021-06-01 20:57:32.607 Done.
1>2021-06-01 20:57:32.607 Done.
1>2021-06-01 20:57:32.607 Done.
1>2021-06-01 20:57:32.607 Done.
1>Done building project "Model.vcxproj".
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========
```



Note: you can build Debug version of the model and run it on cluster, but actual debuging on cluster is far from being trivial.

Using older versions of Visual Studio

OpenM++ tested on current version of Windows 10 and Visual Studio and it is likely works on previous versions too, but it is not tested. If you experiencing an issues with model build please try below recepies.

If you getting link unresolved external symbol errors:

You may get linkage errors if your model `.obj` files incompatible with object files in openM++ library or Microsoft VC++ libraries. For example, build error messages may look like:

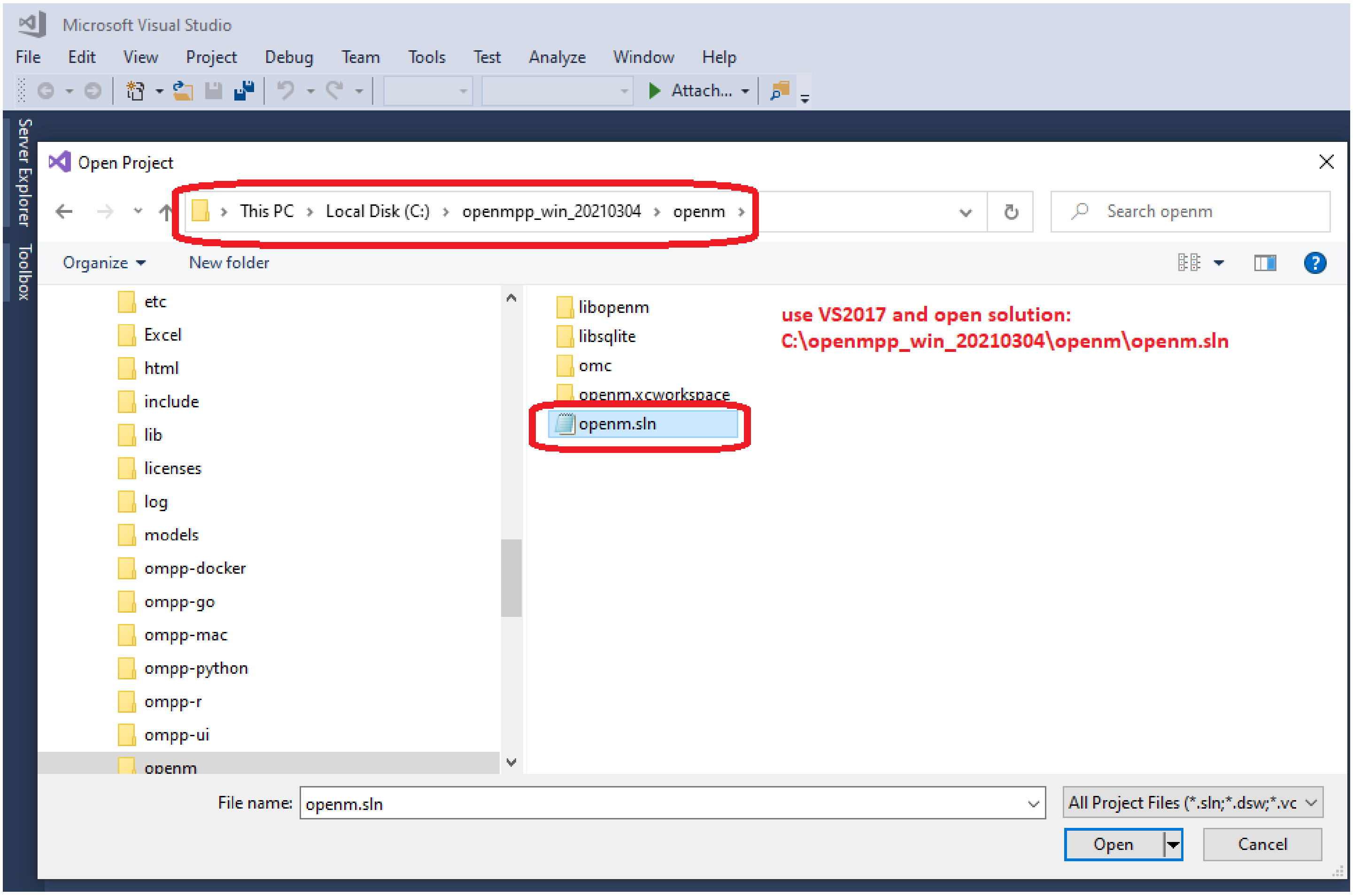
```
1>libopenm.lib(main.obj) : error LNK2001: unresolved external symbol __imp____std_init_once_begin_initialize@16
1>libopenm.lib(main.obj) : error LNK2001: unresolved external symbol __imp____std_init_once_complete@12
1>libopenm.lib(file.obj) : error LNK2001: unresolved external symbol ___std_system_error_allocate_message@8
1>libopenm.lib(file.obj) : error LNK2001: unresolved external symbol ___std_system_error_deallocate_message@4
1>C:\openmpp_win_20210304\models\RiskPaths\ompp\bin\RiskPaths.exe : fatal error LNK1120: 4 unresolved externals
```

1. Clean existing model intermediate files and build model again. Assuming your model directory is `C:\openmpp_win_20210304\models\RiskPaths` then remove following directories:

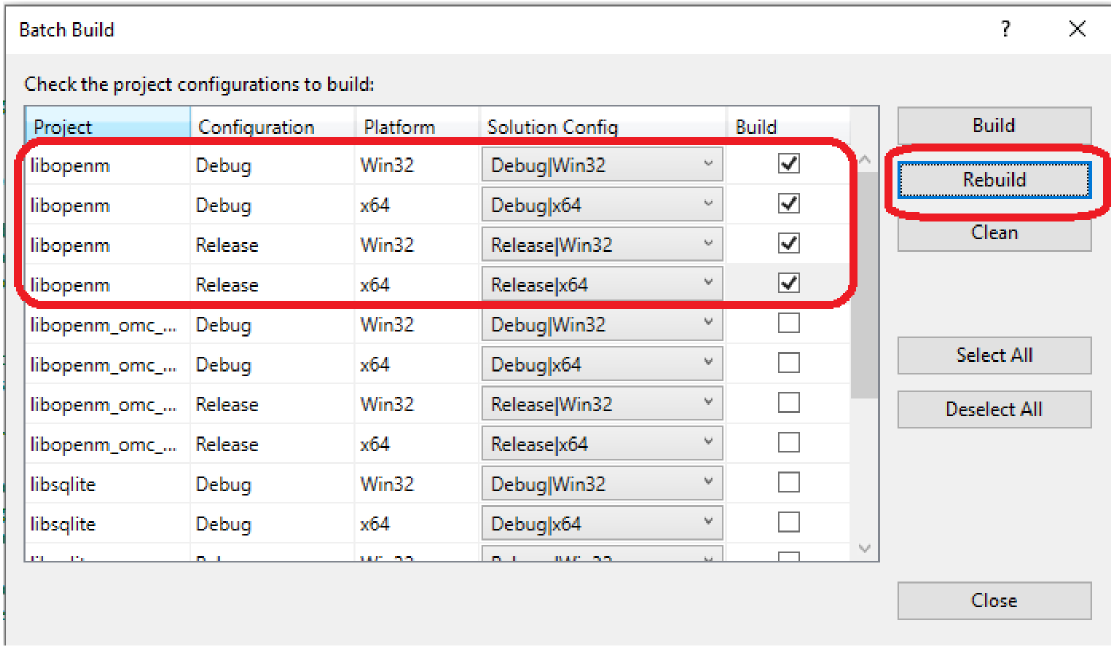
```
C:\openmpp_win_20210304\models\RiskPaths\ompp\bin\
C:\openmpp_win_20210304\models\RiskPaths\ompp\build\
C:\openmpp_win_20210304\models\RiskPaths\ompp\src\
```

2. If you are using Visual Studio 2017 then recepie above may not solve the problem. In that case you need to rebuild `libopenm` openM++ model run-time libarary.

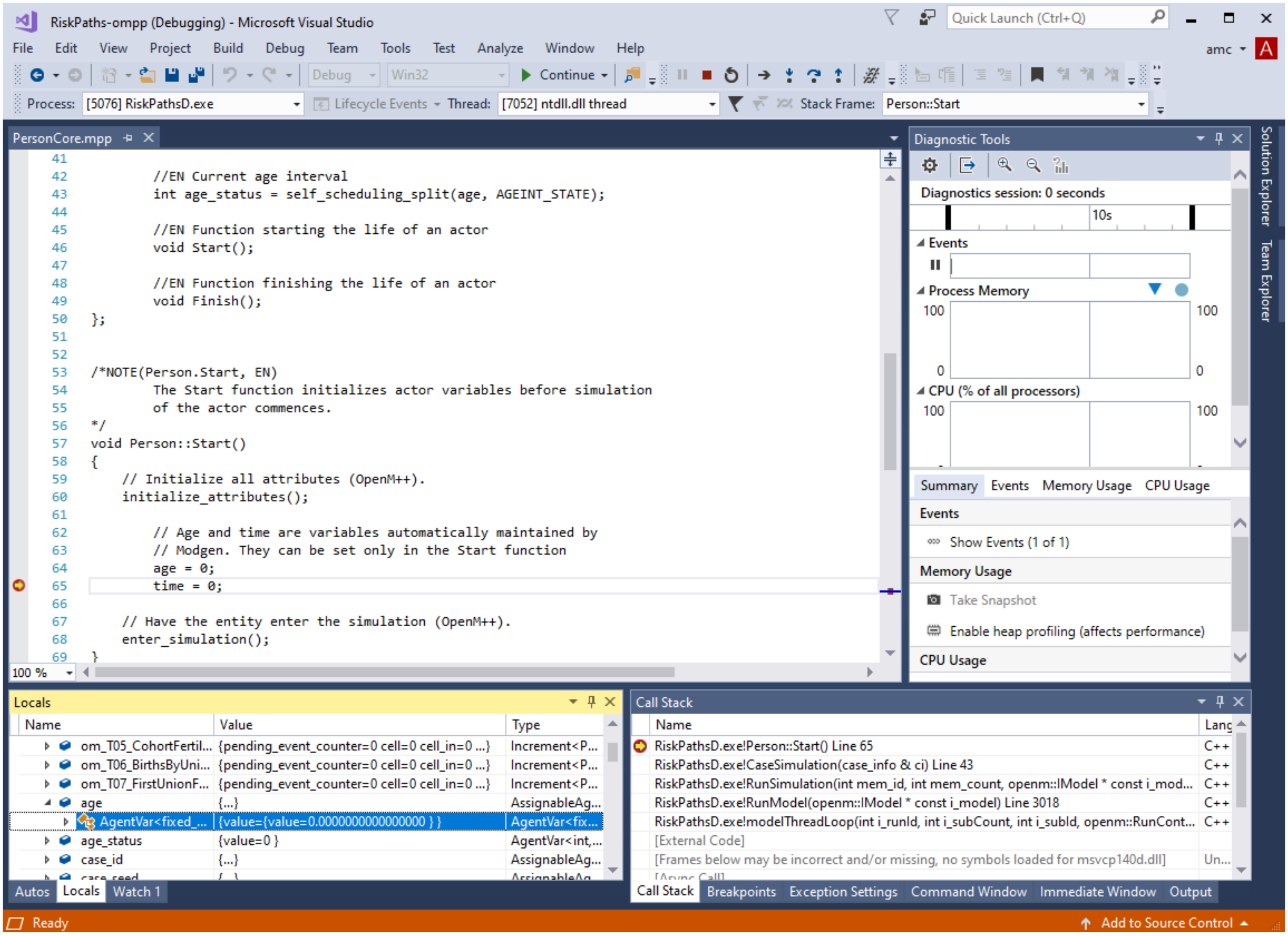
3. Open solution `C:\openmpp_win_20210304\openm\openm.sln` :



- Rebuild `libopenm` library:
  - Visual Studio menu -> Build -> Batch Build...
  - select all `libopenm` projects: `Debug / Release / x64 / Win32`
  - click on Rebuild



- Open your model solution and do rebuild. It is expected to work and you should be able to debug your model even with Visual Studio 2017:



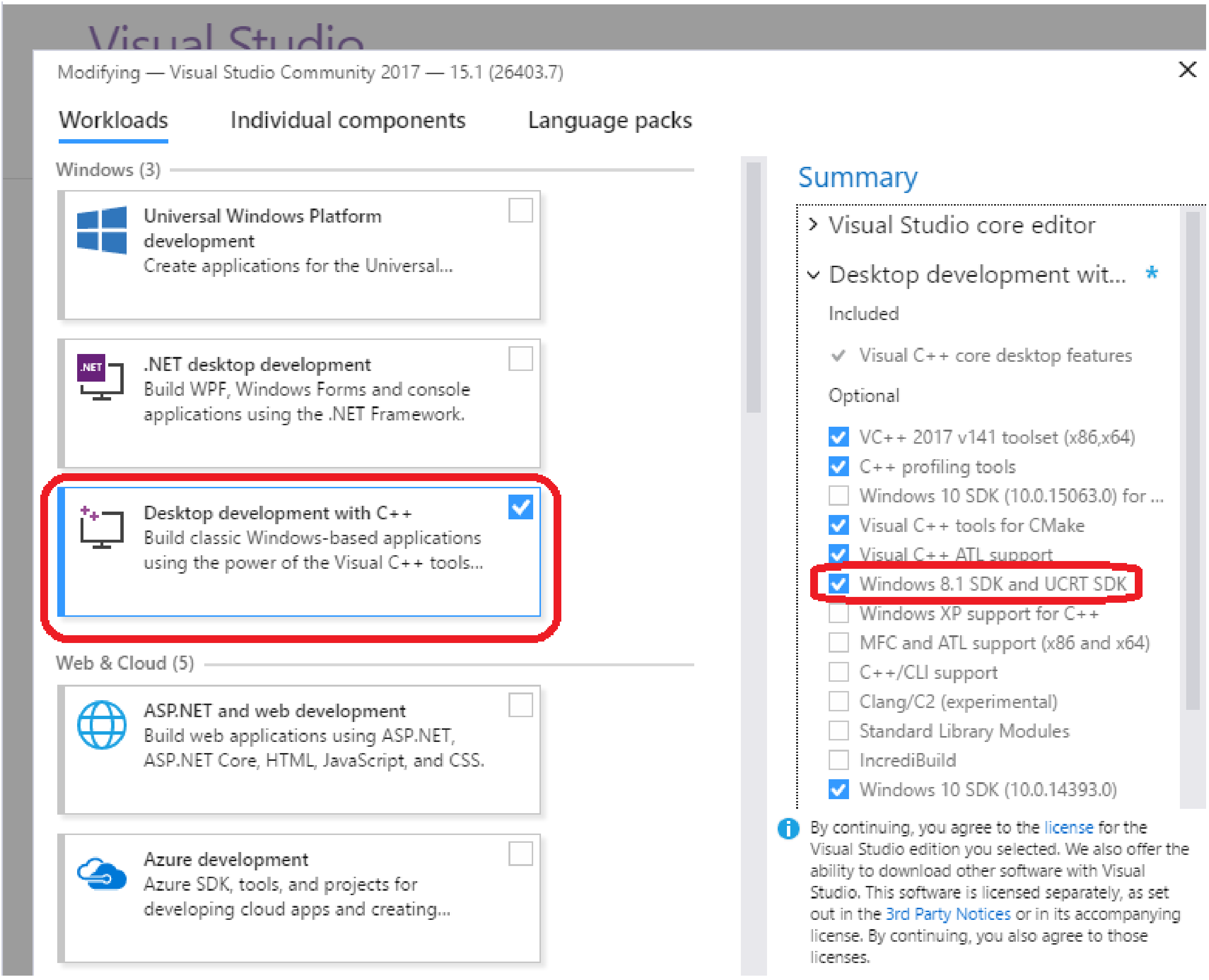
If you getting build error MSB8036:

```
C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\Common7\IDE\VC\VCTargets\Platforms\Win32\PlatformToolsets\v141\Toolset.targets(34,5):
error MSB8036: The Windows SDK version 10.0.14393.0 was not found.
Install the required version of Windows SDK or change the SDK version in the project property pages or by right-clicking the solution and selecting "Retarget solution".
```

then do one of:

- "Retarget solution"

- use Visual Studio 2019
- install Windows 8.1 SDK and UCRT SDK:



Linux: Quick Start for Model Users

Where is OpenM++

- Download:
  - desktop version: binary files and source code openmpp_debian_YYYYMMDD.tar.gz
  - cluster version: binary files and source code openmpp_debian_mpi_YYYYMMDD.tar.gz
  - Docker image to run openM++ models: openmpp/openmpp-run:debian
- Documentation: this wiki

It is recommended to start from desktop version of openM++.

You need to use cluster version of openM++ to run the model on multiple computers in your network, in cloud or HPC cluster environment. OpenM++ is using MPI to run the models on multiple computers. Please check Model Run: How to Run the Model page for more details.

You can use Docker containers to avoid installation of multiple additional components in your host computer. Because all necessary software will be installed in container your host system will be clean.

Run on Linux computer

- download and unpack openM++, i.e.:

```
wget https://github.com/openmpp/main/releases/download/v1.2.0/openmpp_debian_20190508.tar.gz
tar xzf openmpp_debian_20190508.tar.gz
```

- run modelOne model with single subsample on local machine:

```
cd openmpp_debian_20190508/models/bin/
./modelOne
```

```
2017-06-06 19:24:53.0747 modelOne
2017-06-06 19:24:53.0763 Run: 105
2017-06-06 19:24:53.0763 Reading Parameters
2017-06-06 19:24:53.0764 Running Simulation
2017-06-06 19:24:53.0765 Writing Output Tables
2017-06-06 19:24:53.0790 Done.
```

- run modelOne model with 16 subsamples and 4 threads:

```
./modelOne -OpenM.Subvalues 16 -OpenM.Threads 4
```

```
2017-06-06 19:25:38.0721 modelOne
2017-06-06 19:25:38.0735 Run: 106
2017-06-06 19:25:38.0735 Reading Parameters
........................
2017-06-06 19:25:38.0906 Done.
```

- run other models (i.e. NewCaseBased, NewTimeBased, RiskPaths):

```
./NewCaseBased -OpenM.Subvalues 32 -OpenM.Threads 4
```

- run RiskPaths model with new parameter value `CanDie = true` and all other parameter values the same as in previous model run:

```
RiskPaths -Parameter.CanDie true -OpenM.BaseRunId 102
```

```
2020-08-14 17:27:48.574 RiskPaths
2020-08-14 17:27:48.610 Run: 103
2020-08-14 17:27:48.618 Sub-value: 0
2020-08-14 17:27:48.628 member=0 Simulation progress=0% cases=0
........................
2020-08-14 17:27:54.883 Done.
```

- run modelOne to compute modeling task "taskOne":

```
./modelOne -OpenM.Subvalues 16 -OpenM.Threads 4 -OpenM.TaskName taskOne
```

```
2017-06-06 19:27:08.0401 modelOne
2017-06-06 19:27:08.0421 Run: 107
2017-06-06 19:27:08.0421 Reading Parameters
........................
2017-06-06 19:27:08.0593 Run: 108
2017-06-06 19:27:08.0593 Reading Parameters
........................
2017-06-06 19:27:08.0704 Writing Output Tables
2017-06-06 19:27:08.0812 Done.
```

- in case if previous model run fail, for example, due to power outage, then it can be "restarted":

```
./modelOne -OpenM.RestartRunId 1234
```

  output may vary depending on the stage where previous modelOne run failed, but still similar to above.

Run on multiple computers over network, in HPC cluster or cloud

- make sure you have MPI run-time installed and ready to use. For example, on RedHat you may need to load it by following commands:

```
module load mpi/openmpi-x86_64
```

  As an alternative to MPI installation you can pull Docker image `docker pull openmpp/openmpp-run:debian` to run models inside the container (see below).

- download and unpack cluster version of openM++, i.e.:

```
wget https://github.com/openmpp/main/releases/download/v1.2.0/openmpp_debian_mpi_20190508.tar.gz
tar xzf openmpp_debian_mpi_20190508.tar.gz
```

  please notice name of cluster version archive has ***mpi*** in it, i.e. `openmpp_debian_mpi_20190508.tar.gz`

- run modelOne model with single subsample on local machine:

```
cd openmpp_debian_mpi_20190508/models/bin/
./modelOne_mpi
```

```
2017-06-06 19:30:52.0690 Run: 105
2017-06-06 19:30:52.0690 Reading Parameters
2017-06-06 19:30:52.0691 Running Simulation
2017-06-06 19:30:52.0691 Writing Output Tables
2017-06-06 19:30:52.0716 Done.
```

- run two instances of modelOne to compute 16 subsamples and 4 threads:

```
mpiexec -n 2 modelOne_mpi -OpenM.Subvalues 16 -OpenM.Threads 4
```

```
2017-06-06 19:43:01.0486 modelOne
2017-06-06 19:43:01.0487 modelOne
2017-06-06 19:43:01.0742 Parallel run of 2 modeling processes, 4 thread(s) each
2017-06-06 19:43:01.0750 Run: 106
2017-06-06 19:43:01.0750 Reading Parameters
2017-06-06 19:43:01.0750 Run: 106
2017-06-06 19:43:01.0750 Reading Parameters
..........
2017-06-06 19:43:01.0800 Writing Output Tables
2017-06-06 19:43:01.0878 Done.
2017-06-06 19:43:01.0880 Done.
```

- run other models (i.e. NewCaseBased, NewTimeBased, RiskPaths):

```
mpiexec -n 8 NewCaseBased_mpi -OpenM.Subvalues 64 -OpenM.Threads 4
```

It is recommended to install SLURM or Torque to simplify your computational resources management rather than using `mpiexec` as above. It is also possible to use Google Cloud, Amazon or even Microsoft Azure cloud where compute nodes available for you on demand.

Run models using Docker container

- make sure you have Docker installed, for example, on Ubuntu: `sudo apt-get install docker`.
- pull Docker image:

  ```
  docker pull openmpp/openmpp-run:debian
  ```

- image build for user `ompp, UID=1999, GID=1999` and you may need to do one of:
  - add same user `ompp, UID=1999, GID=1999` to your host system and login as user `ompp`
  - or as shown below use environment variables `OMPP_*` to map your current user name, UID, GID, HOME to container user
- download and unpack cluster version of openM++, i.e.:

  ```
  wget https://github.com/openmpp/main/releases/download/v1.2.0/openmpp_debian_mpi_20200621.tar.gz
  tar xzf openmpp_debian_mpi_20200621.tar.gz
  ```

  please notice name of cluster version archive has ***mpi*** in it, i.e. `openmpp_debian_mpi_20200621.tar.gz`
- run modelOne model with single subsample on local machine:

  ```
  docker run \
  -v $HOME/models/bin:/home/models \
  -e OMPP_USER=models -e OMPP_GROUP=models -e OMPP_UID=$UID -e OMPP_GID=`id -g` \
  openmpp/openmpp-run:debian \
  ./modelOne_mpi
  ```

  ```
  2017-06-06 19:30:52.0690 Run: 105
  2017-06-06 19:30:52.0690 Reading Parameters
  2017-06-06 19:30:52.0691 Running Simulation
  2017-06-06 19:30:52.0691 Writing Output Tables
  2017-06-06 19:30:52.0716 Done.
  ```

  For explanation of:

  ```
  -v $HOME/models/bin:/home/models \
  -e OMPP_USER=models -e OMPP_GROUP=models -e OMPP_UID=$UID -e OMPP_GID=`id -g` \
  ```

  please take a look at User, group, home_directory topic.
- run two instances of modelOne to compute 16 subsamples and 4 threads:

  ```
  docker run \
  -v $HOME/models/bin:/home/models \
  -e OMPP_USER=models -e OMPP_GROUP=models -e OMPP_UID=$UID -e OMPP_GID=`id -g` \
  openmpp/openmpp-run:debian \
  mpiexec -n 2 modelOne_mpi -OpenM.Subvalues 16 -OpenM.Threads 4
  ```

  ```
  2017-06-06 19:43:01.0486 modelOne
  2017-06-06 19:43:01.0487 modelOne
  2017-06-06 19:43:01.0742 Parallel run of 2 modeling processes, 4 thread(s) each
  2017-06-06 19:43:01.0750 Run: 106
  2017-06-06 19:43:01.0750 Reading Parameters
  2017-06-06 19:43:01.0750 Run: 106
  2017-06-06 19:43:01.0750 Reading Parameters
  ..........
  2017-06-06 19:43:01.0800 Writing Output Tables
  2017-06-06 19:43:01.0878 Done.
  2017-06-06 19:43:01.0880 Done.
  ```

- run other models (i.e. NewCaseBased, NewTimeBased, RiskPaths):

```
docker run \
....user, UID, GID, HOME.... \
openmpp/openmpp-run:debian \
mpiexec -n 8 NewCaseBased_mpi -OpenM.Subvalues 64 -OpenM.Threads 4
```

Linux: Quick Start for Model Developers

Where is OpenM++

- Download:
  - desktop version: binary files and source code openmpp_debian_YYYYMMDD.tar.gz
  - cluster version: binary files and source code openmpp_debian_mpi_YYYYMMDD.tar.gz
- How to: create and debug models on Linux

It is recommended to start from desktop version of openM++.

You need to use cluster version of openM++ to run the model on multiple computers in your network, in cloud or HPC cluster environment. OpenM++ is using MPI to run the models on multiple computers. Please check Model Run: How to Run the Model page for more details.

Build on Linux

Tested platforms:

- Debian 10, MX Linux 19, Ubuntu 20.04, RedHat 8
- g++ >= 8.3
- (optional) MPI, i.e.: OpenMPI >= 3.1 or MPICH (other MPI implementations expected to work but not tested)
- (optional) OpenMPI >= 4.0 on RedHat >= 8.3 (OpenMPI was broken on RedHat 8.1)

It is also occasionally tested on openSUSE, Mint, Manjaro, Solus and others.

It is not supported, but may also work on older versions, for example Ubuntu 18.04.

Build on Ubuntu 20.04

There is a minor incompatibility of shared libraries between Ubuntu 20.04 and Debian 10. As result you need to rebuild our model run-time libraries before building you own model:

- download and unpack openM++ into any folder:

```
wget https://github.com/openmpp/main/releases/download/v1.8.6/openmpp_debian_20210415.tar.gz
tar xzf openmpp_debian_20210415.tar.gz
```

- rebuild model run-time libraries:

```
cd openmpp_debian_20210415/openm
wget https://github.com/openmpp/main/releases/download/v1.8.6/openmpp_debian_20210415.tar.gz
tar xzf openmpp_debian_20210415.tar.gz
```

Build debug version of the model

You can use any of test models makefile, except of modelOne, as starting point to develop your own model. Below we are using NewCaseBased model as example.

To build and run **debug version** of the model use desktop (non-MPI) version of openM++:

- check your g++ --version:

```
g++ (Debian 8.3.0-6) 8.3.0
g++ (Ubuntu 9.3.0-10ubuntu2) 9.3.0
g++ (GCC) 8.3.1 20191121 (Red Hat 8.3.1-5)
```

- download and unpack openM++

```
wget https://github.com/openmpp/main/releases/download/v1.8.3/openmpp_debian_20210304.tar.gz
tar xzf openmpp_debian_20210304.tar.gz
```

- build debug version of NewCaseBased model and "publish" it ("publish" do create NewCaseBased.sqlite database with default input data set)

```
cd openmpp_debian_20210304/models/NewCaseBased/
make all publish
```

- run the model

```
cd ompp-linux/bin
./NewCaseBasedD
```

```
2017-06-06 19:59:12.0429 NewCaseBased
2017-06-06 19:59:12.0449 Run: 103
2017-06-06 19:59:12.0449 Get fixed and missing parameters
2017-06-06 19:59:12.0449 Get scenario parameters
2017-06-06 19:59:12.0450 Sub-value 0
2017-06-06 19:59:12.0450 compute derived parameters
2017-06-06 19:59:12.0450 Initialize invariant entity data
2017-06-06 19:59:12.0450 Member=0 simulation progress=0%
..........
2017-06-06 19:59:12.0505 member=0 write output tables - finish
2017-06-06 19:59:12.0508 Writing Output Tables Expressions
2017-06-06 19:59:12.0520 Done.
```

Build release version of the model

Make executable, "publish" and run NewCaseBased test model:

```
cd openmpp_debian_20210304/models/NewCaseBased/
make RELEASE=1 clean-all
make RELEASE=1 all publish
cd ompp-linux/bin
./NewCaseBased
```

Rebuild all test models

Make executables, "publish" (create model.sqlite database file) and run all test models:

```
cd openmpp_debian_20210304/models/
make RELEASE=1 clean-all
make RELEASE=1 all publish run publish-all
```

results are in `openmpp_debian_20210304/models/bin` directory

OM_ROOT: How to separate model folder and openM++ release folder

If you want to keep model development folder(s) outside of openM++ release directory then set `OM_ROOT` environment variable to specify openM++ release location. For example if your model is in `$HOME/my-models/BestModel` then to build it do any of:

```
cd my-models/BestModel
OM_ROOT=openmpp_debian_20210304 make all publish run
```

Or edit `$HOME/my-models/BestModel/makefile` to set `OM_ROOT` :

```
ifndef OM_ROOT
  OM_ROOT = $(HOME)/openmpp_debian_20210304
endif
```

Or add `export OM_ROOT=$HOME/openmpp_debian_20210304` into your `.bash_profile`

Build cluster version of the model to run on multiple computers over network

Make sure you have MPI installed and configured. For example, on RedHat you may need to load MPI module: `module load mpi/openmpi-x86_64`

- download and unpack cluster version of openM++, i.e.:

```
wget https://github.com/openmpp/main/releases/download/v1.8.3/openmpp_debian_mpi_20210304.tar.gz
tar xzf openmpp_debian_mpi_20210304.tar.gz
```

please notice name of cluster version archive has ***mpi*** in it, i.e. `openmpp_debian_mpi_20210304.tar.gz`

- make executable and "publish" (create model.sqlite database file) of NewCaseBased test model:

```
cd openmpp_debian_mpi_20210304/models/NewCaseBased/
make RELEASE=1 OM_MSG_USE=MPI all publish
```

- run 3 instances of NewCaseBased on 3 hosts to compute 16 subsamples using 4 threads

```
cd ompp-linux/bin
mpirun -n 3 -H omm,om1,om2 NewCaseBased_mpi -OpenM.Subvalues 16 -OpenM.Threads 4
```

```
2017-06-06 20:15:12.0050 NewCaseBased
2017-06-06 20:15:12.0173 NewCaseBased
2017-06-06 20:15:12.0200 NewCaseBased
2017-06-06 20:15:13.0148 Parallel run of 3 modeling processes, 4 thread(s) each
2017-06-06 20:15:13.0162 Run: 102
2017-06-06 20:15:13.0163 Get fixed and missing parameters
2017-06-06 20:15:13.0163 Get scenario parameters
2017-06-06 20:15:13.0164 compute derived parameters
2017-06-06 20:15:13.0164 Initialize invariant entity data
2017-06-06 20:15:13.0161 Run: 102
..........
2017-06-06 20:15:13.0224 member=0 write output tables - finish
2017-06-06 20:15:13.0354 Done.
2017-06-06 20:15:13.0352 Done.
2017-06-06 20:15:13.0353 Done.
```

You can use any of test models makefile, except of modelOne, as starting point to develop your own model.

MacOS: Quick Start for Model Users

Where is OpenM++

- Download latest binary files and source code: openmpp_mac_YYYYMMDD.tar.gz
- Documentation: this wiki

You can have multiple versions of openM++ installed on your computer. OpenM++ distributed as tar.gz archive, you can unpack into any directory and it is ready to use. In the documentation that directory called OM_ROOT.

OpenM++ does not update any system shared resources and you can remove it any time by simply deleting openM++ directory.

It is possible to run openM++ models:

- from terminal command line as described below
- using openM++ UI on your local computer: UI: openM++ user interface
- from Xcode model debug session: MacOS: Quick Start for Model Developers

On Linux and/or Windows you also can run model in cloud or on high perfomance cluster (HPC). Please also check Model Run: How to Run the Model page for more details.

Run openM++ models from terminal command line

- download and unpack openM++ using Safari or, for example, curl:

```
curl -L -o om.tar.gz https://github.com/openmpp/main/releases/download/v1.6.0/openmpp_mac_20200621.tar.gz
tar xzf om.tar.gz
```

- run modelOne model with single sub-sample on local machine:

```
cd openmpp_mac_20200621/models/bin/
./modelOne
```

```
2017-06-06 19:24:53.0747 modelOne
2017-06-06 19:24:53.0763 Run: 105
2017-06-06 19:24:53.0763 Reading Parameters
2017-06-06 19:24:53.0764 Running Simulation
2017-06-06 19:24:53.0765 Writing Output Tables
2017-06-06 19:24:53.0790 Done.
```

- run modelOne model with 16 sub-samples and 4 threads:

```
./modelOne -OpenM.Subvalues 16 -OpenM.Threads 4
```

```
2017-06-06 19:25:38.0721 modelOne
2017-06-06 19:25:38.0735 Run: 106
2017-06-06 19:25:38.0735 Reading Parameters
........................
2017-06-06 19:25:38.0906 Done.
```

- run other models (i.e. NewCaseBased, NewTimeBased, RiskPaths):

```
./NewCaseBased -OpenM.Subvalues 32 -OpenM.Threads 4
```

- run RiskPaths model with new parameter value `CanDie = true` and all other parameter values the same as in previous model run:

```
RiskPaths -Parameter.CanDie true -OpenM.BaseRunId 102
```

```
2020-08-14 17:27:48.574 RiskPaths
2020-08-14 17:27:48.610 Run: 103
2020-08-14 17:27:48.618 Sub-value: 0
2020-08-14 17:27:48.628 member=0 Simulation progress=0% cases=0
........................
2020-08-14 17:27:54.883 Done.
```

- run modelOne to compute modeling task "taskOne":

```
./modelOne -OpenM.Subvalues 16 -OpenM.Threads 4 -OpenM.TaskName taskOne
```

```
2017-06-06 19:27:08.0401 modelOne
2017-06-06 19:27:08.0421 Run: 107
2017-06-06 19:27:08.0421 Reading Parameters
........................
2017-06-06 19:27:08.0593 Run: 108
2017-06-06 19:27:08.0593 Reading Parameters
........................
2017-06-06 19:27:08.0704 Writing Output Tables
2017-06-06 19:27:08.0812 Done.
```

- in case if previous model run fail, for example, due to power outage, then it can be "restarted":

```
./modelOne -OpenM.RestartRunId 1234
```

  output may vary depending on the stage where previous modelOne run failed, but still similar to above.

MacOS: Quick Start for Model Developers

Where is OpenM++

- Download: latest binary files and source code
- How to: create and debug models on MacOS

Also, please check Model Run: How to Run the Model page for more details.

Prerequisites

- Tested on: MacOS 10.15 Catalina And Big Sur >= 11.1.
- Install Xcode and command line developer tools, if not installed already by Xcode: `xcode-select --install`.
- (optional) Install Visual Studio Code for cross-platform development: https://code.visualstudio.com/docs/?dv=osx
- Check if clang, make and sqlite3 are installed on your computer:

```
g++ --version
....
Apple clang version 11.0.0 (clang-1100.0.33.12)

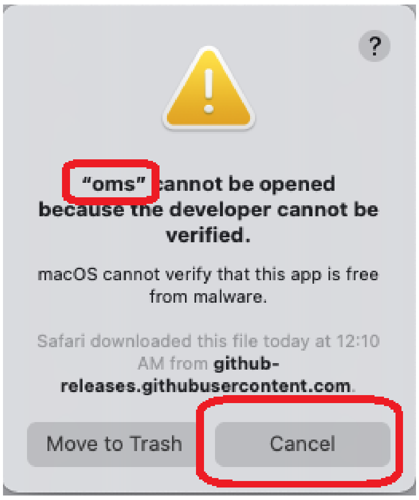
make --version
....
GNU Make 3.81

sqlite3 --version
....
3.28.0 2019-04-15 14:49:49
```

MacOS security issue

Make sure you are using tight security settings on your Mac and antivirus software, if necessary. We are trying our best to keep development machines clean, but cannot provide any guarantee.

On Big Sur it is very likely to get an security error when you are trying to run any downloaded executable:



- please reply "Cancel" to that question (click "Cancel" button).
- remove quarantine attribute from openM++ installation directory, for example:

```
xattr -r -d com.apple.quarantine ~/openmpp_mac_20200621
```

Build debug version of the model from terminal command line

You can use any of test models makefile, except of modelOne, as starting point to develop your own model. Below we are using NewCaseBased model as example.

To build and run **debug version** of the model:

- download and unpack latest openM++ release using Safari or curl:

```
curl -L -o om.tar.gz https://github.com/openmpp/main/releases/download/v1.6.0/openmpp_mac_20200621.tar.gz
tar -xzf om.tar.gz
```

- remove quarantine attribute from openM++ installation directory:

```
xattr -r -d com.apple.quarantine openmpp_mac_20200621
```

- build debug version of NewCaseBased model and "publish" it ("publish" do create NewCaseBased.sqlite database with default input data set)

```
cd openmpp_mac_20200621/models/NewCaseBased/
make all publish
```

- run the model

```
cd ompp-mac/bin
./NewCaseBasedD
```

```
2017-06-06 19:59:12.0429 NewCaseBased
2017-06-06 19:59:12.0449 Run: 103
2017-06-06 19:59:12.0449 Get fixed and missing parameters
2017-06-06 19:59:12.0449 Get scenario parameters
2017-06-06 19:59:12.0450 Sub-value 0
2017-06-06 19:59:12.0450 compute derived parameters
2017-06-06 19:59:12.0450 Initialize invariant entity data
2017-06-06 19:59:12.0450 Member=0 simulation progress=0%
..........
2017-06-06 19:59:12.0505 member=0 write output tables - finish
2017-06-06 19:59:12.0508 Writing Output Tables Expressions
2017-06-06 19:59:12.0520 Done.
```

- you can also build and run the model using make:

```
make all publish run
..........
2017-06-06 19:59:12.0429 NewCaseBased
2017-06-06 19:59:12.0449 Run: 103
..........
2017-06-06 19:59:12.0508 Writing Output Tables Expressions
2017-06-06 19:59:12.0520 Done.
```

Build release version of the model from terminal command line

Make executable, "publish" and run NewCaseBased test model:

```
cd openmpp_mac_20200621/models/NewCaseBased/
make RELEASE=1 clean-all
make RELEASE=1 all publish
cd ompp-mac/bin
./NewCaseBased
```

Rebuild all test models

Make executables, "publish" (create model.sqlite database file) and run all test models:

```
cd openmpp_mac_20200621/models/
make RELEASE=1 clean-all
make RELEASE=1 all publish run publish-all
```

results are in `openmpp_mac_20200621/models/bin` directory

OM_ROOT: How to separate model folder and openM++ release folder

If you want to keep model development folder(s) outside of openM++ release directory then set `OM_ROOT` environment variable to specify openM++ release location. For example if your model is in `$HOME/my-models/BestModel` then to build it do any of:

```
cd my-models/BestModel
OM_ROOT=openmpp_mac_20200621 make all publish run
```

Or edit `$HOME/my-models/BestModel/makefile` to set `OM_ROOT` :

```
ifndef OM_ROOT
  OM_ROOT = $(HOME)/openmpp_mac_20200621
endif
```

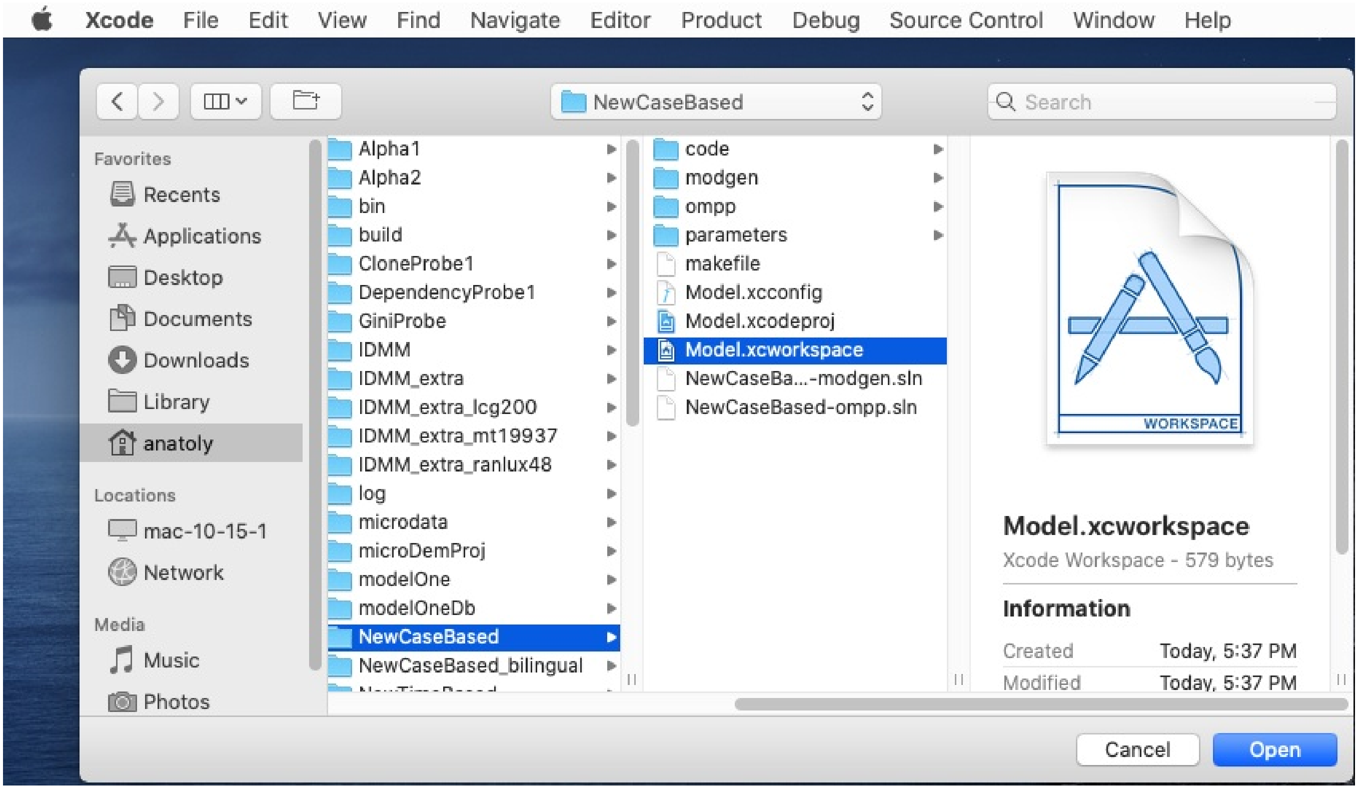
Or add `export OM_ROOT=$HOME/openmpp_mac_20200621` into your `.zprofile`

Build openM++ sample model using Xcode

Download and unpack latest openM++ release using Safari or curl:

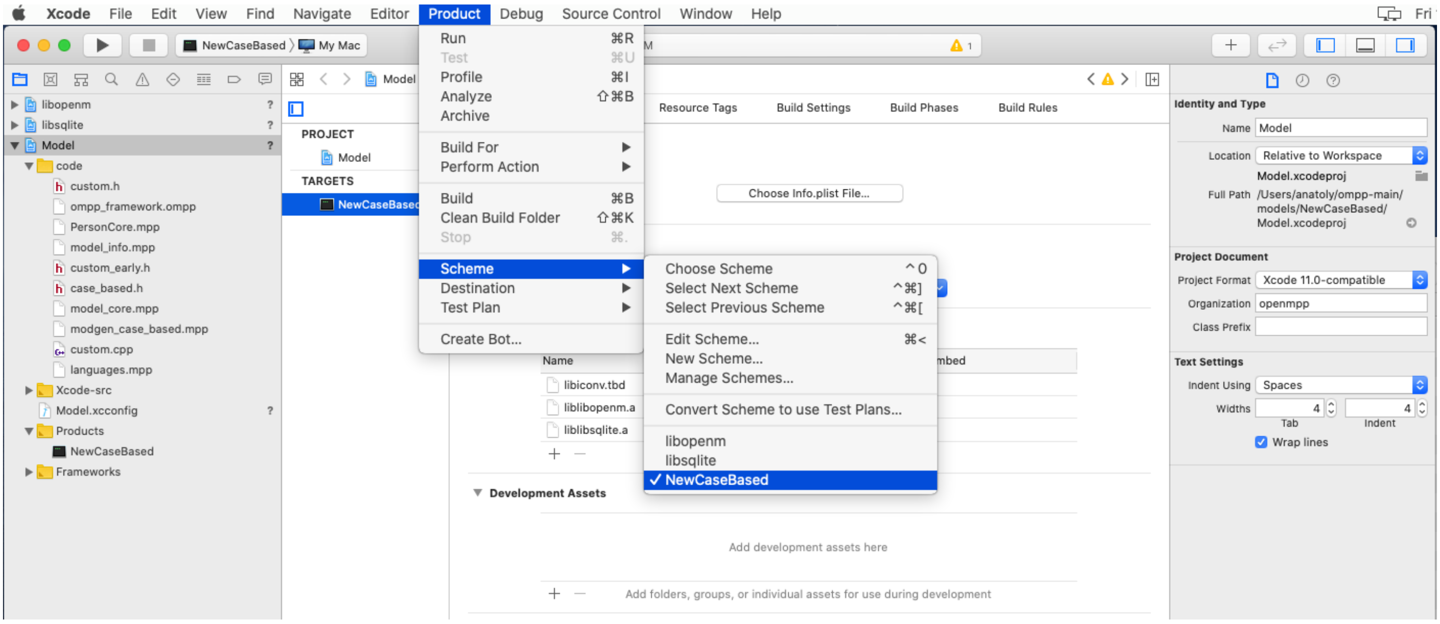
```
curl -L -o om.tar.gz https://github.com/openmpp/main/releases/download/v1.6.0/openmpp_mac_20200621.tar.gz
tar xzf om.tar.gz
```

Start Xcode and open any example model workspace, for example: `~/openmpp_mac_20200621/models/NewCaseBased/Model.xcworkspace`



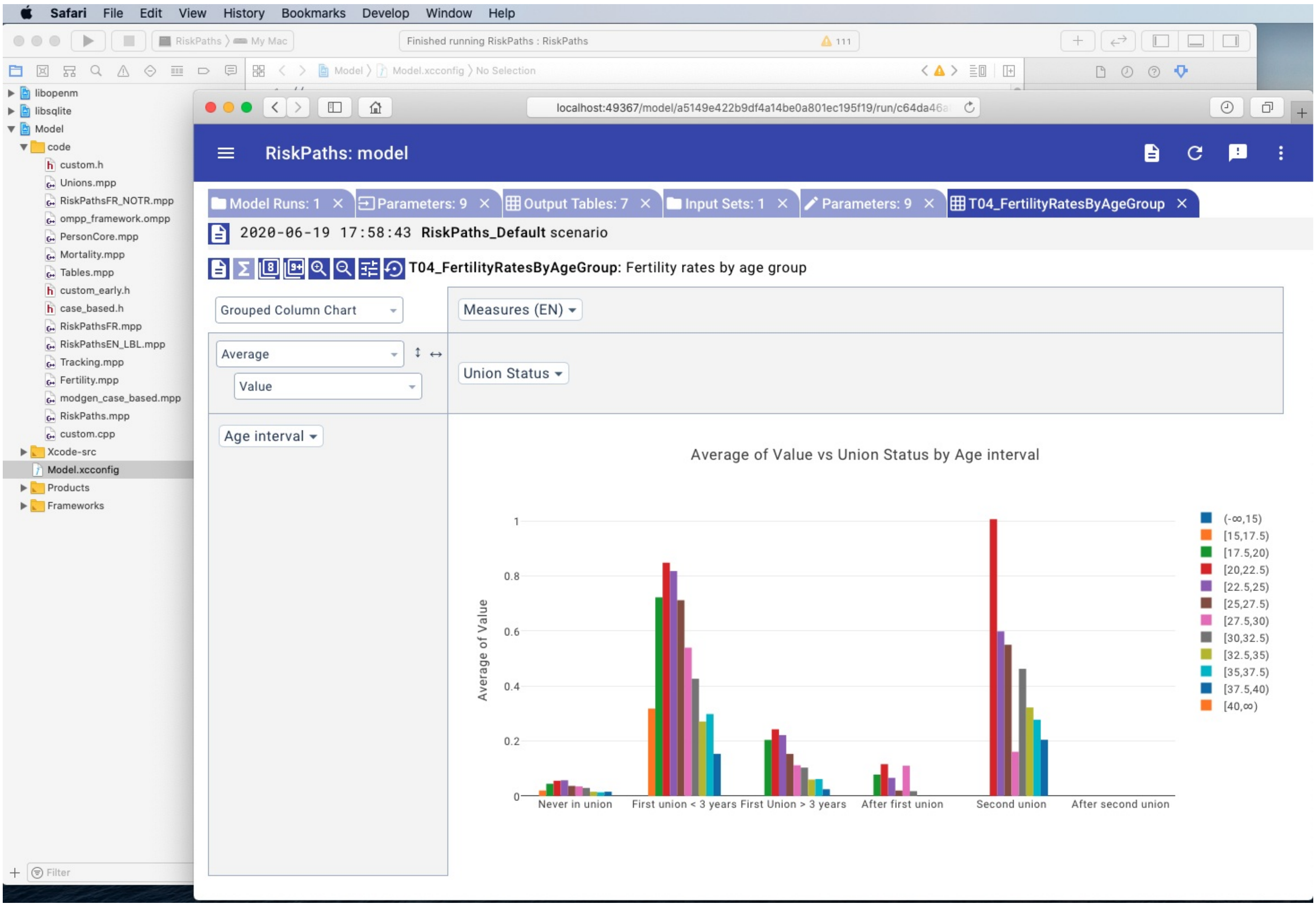
Use menu to select Product -> Scheme -> NewCaseBased:

Known issue: Xcode UI may not update check mark on selected scheme. To fix it go to Product -> Scheme -> Manage Schemes and use mouse to drag any scheme to move it up or down.



Build, debug and run openM++ example model(s) using Xcode.

Open model UI (beta) to update parameters, run the model and view results. To start model UI after build completed please change Model.xcconfig variable START_OMPP_UI to "1" or "true" or "yes" (case-sensitive). Please see details at: Start model UI on MacOS from Xcode



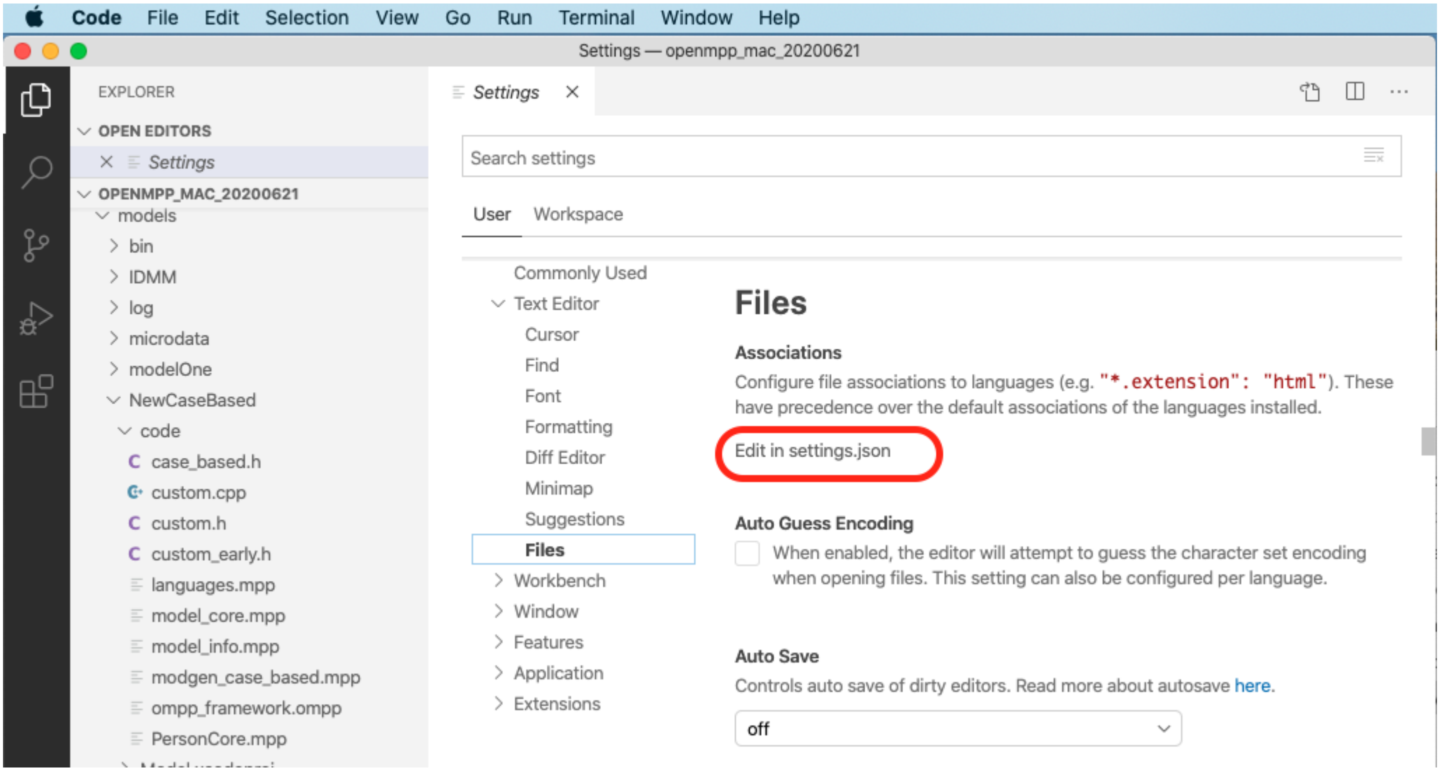
Install VSCode

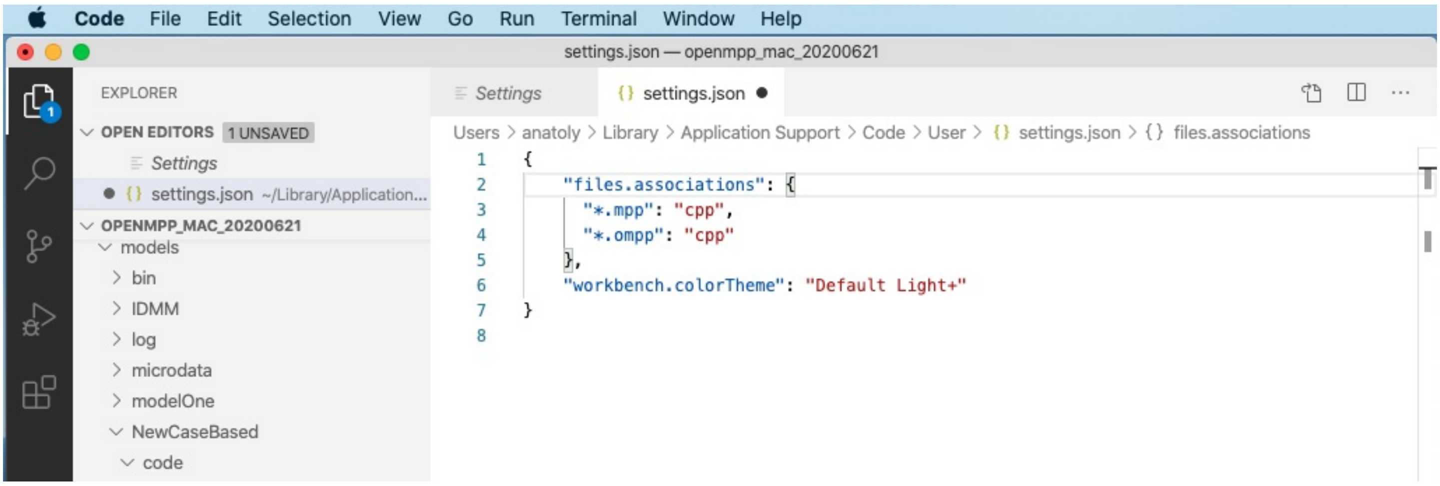
It is convenient to use the same Visual Studio Code IDE if you need to develop on openM+++ models on multiple platforms (Linux, MacOS and Windows). To install VSCode on MacOS and configure for openM++ development do following:

- Download it from: https://code.visualstudio.com/docs/?dv=osx
- Start Visual Studio Code.app and install extension `ms-vscode.cpptools` : C/C++ for Visual Studio Code (Microsoft)

- Define *.ompp and* .mpp file extensions as c++ files by using menu: Code -> Preferences -> Settings -> Text Editor -> Files -> Associations -> Edit in settings.json:

```
{
  "files.associations": {
    "*.mpp": "cpp",
    "*.ompp": "cpp"
  }
}
```





Model Run: How to Run the Model

OpenM++ model run overview

It is recommended to start from single desktop version of openM++.

OpenM++ models can be run on Windows and Linux platforms, on single desktop computer, on multiple computers over network, in HPC cluster or cloud environment (i.e. Google Cloud, Microsoft Azure, Amazon,...).

You need to use cluster version of openM++ to run the model on multiple computers in your network, in cloud or HPC cluster environment. OpenM++ is using MPI to run the models on multiple computers.

By default openM++ model runs with one sub-value and in single thread, which is convenient to debug or study your model. There are following options to run openM++ model:

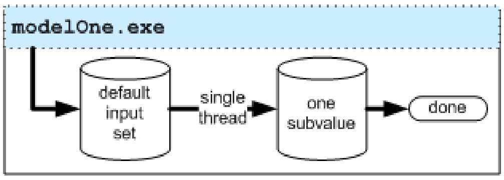
- "default" run: one sub-value and single thread
- "desktop" run: multiple sub-values and multiple threads
- "restart" run: finish model run after previous failure (i.e. power outage)
- "task" run: multiple input sets of data (a.k.a. multiple "scenarios" in Modgen), multiple sub-values and threads
- "cluster" run: multiple sub-values, threads and model process instances runs on LAN or cloud (**required MPI**)
- "cluster task" run: same as "cluster" plus multiple input sets of data (**required MPI**)

Please also check Model Run: How model finds input parameters for more details.

Sub-values: sub-samples, members, replicas

Following terms: "simulation member", "replica", "sub-sample" are often used in micro-simulation conversations interchangeably, depending on context. To avoid terminology discussion openM++ uses "sub-value" as equivalent of all above and some older pages of that wiki may contain "sub-sample" in that case.

Default run: simplest



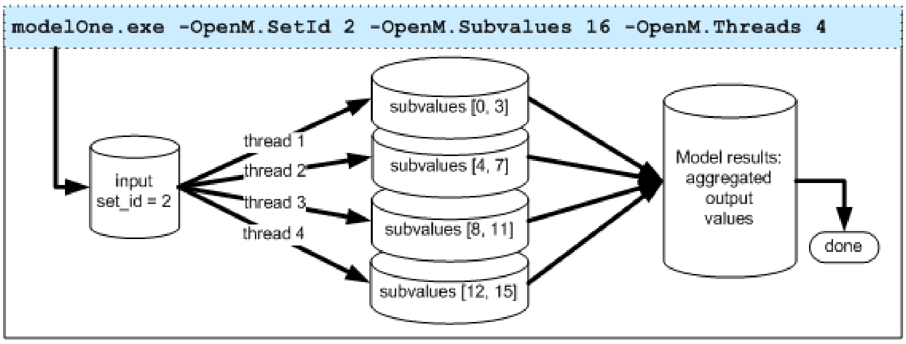
If no any options specified to run the model then

- all parameters are from default input data set
- single thread is used for modeling
- only one sub-value calculated

  ```
  modelOne.exe
  ```

  It is most simple way to debug your model.

Desktop run: model run on single computer



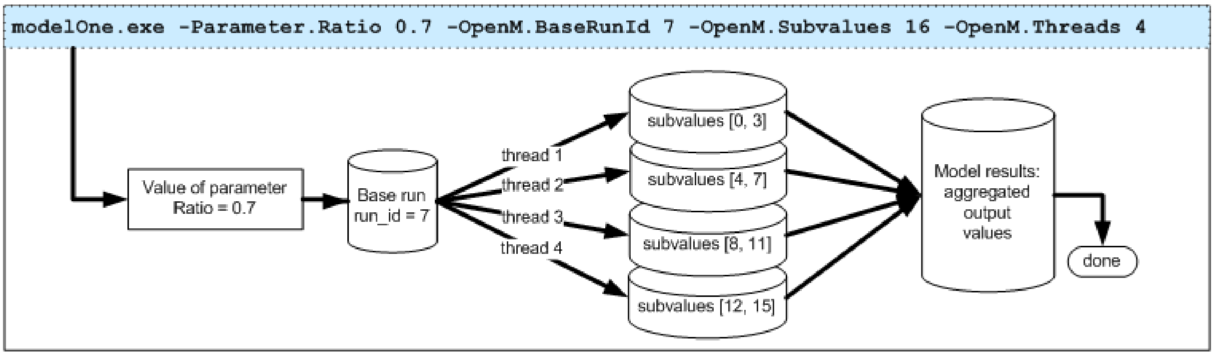
If only single computer available then

- user can specify which set of input data to use (by set name or id)
- number of sub-values to calculate
- number of modeling threads to use

```
modelOne.exe -OpenM.SetName modelOne -OpenM.SubValues 16 -OpenM.Threads 4
```

After model run completed user can repeat it with modified parameter(s) values:

```
model.exe -Parameter.Ratio 0.7 -OpenM.BaseRunId 7 -OpenM.SubValues 16 -OpenM.Threads 4
```



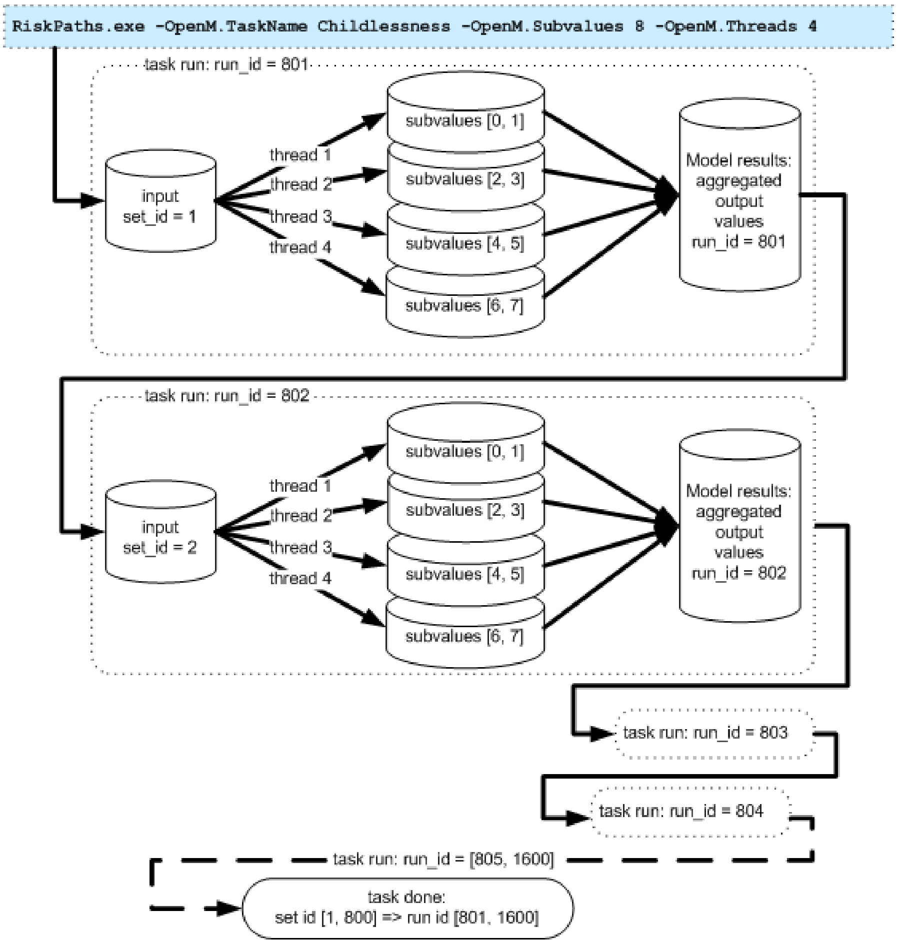
Command above will run the model with new value for parameter `Ratio = 0.7` and use the rest of parameters from previous model run (a.k.a. "base" run). Base run can be identified by run id, which is 7 in example above, by run digest or run name. Please see Model Run: How model finds input parameters for more details.

Restart run: finish model run after previous failure

If previous model run was not completed (i.e. due to power failure or insufficient disk space) you can restart it by specifying run id:

```
modelOne.exe -OpenM.RestartRunId 11
```

Task run: multiple sets of input data



Modeling task consists of multiple sets of input data and can be run in batch mode. For example, it is make sense to create modeling task to Run RiskPaths model from R with 800 sets of input data to study Childlessness by varying

- Age baseline for first union formation
- Relative risks of union status on first pregnancy

```
RiskPaths.exe -OpenM.TaskName Childlessness -OpenM.SubValues 8 -OpenM.Threads 4
```

Run of such modeling task will read 800 input sets with set id [1, 800] and produce 800 model run outputs with run id [801, 1600] respectively.

Dynamic task run: wait for input data

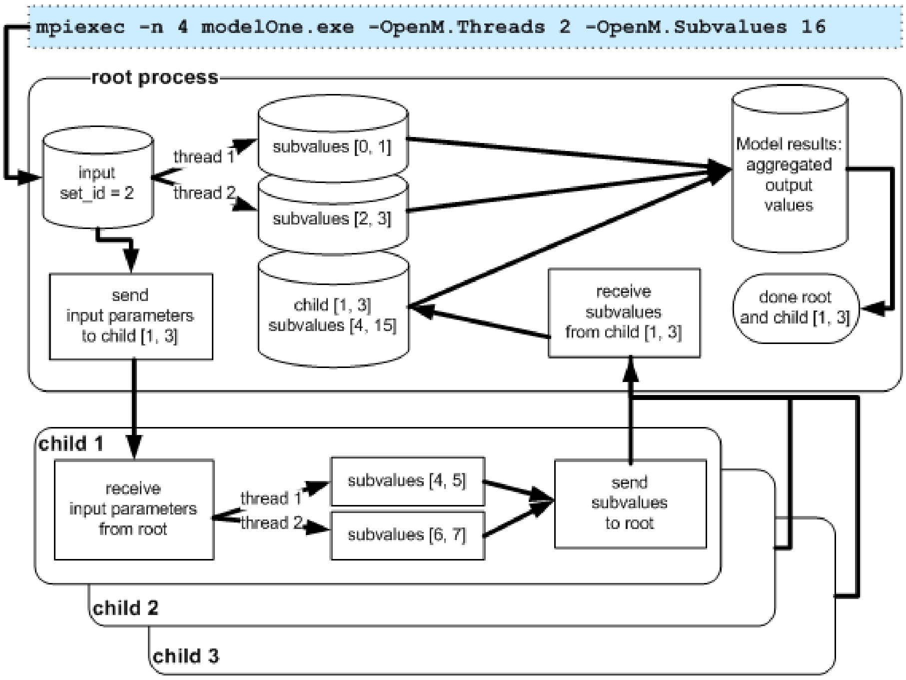
It is possible to append new sets of input data to the task as it runs. That allow you to use some optimization methods rather than simply calculate all possible combinations of input parameters. In that case modeling task does not completed automatically but wait for external "task can be completed" signal. For example:

```
#
# pseudo script to run RiskPaths and find optimal solution for Childlessness problem
# you can use R or any other tools of your choice
#
# # create Childlessness task
# # run loop until you satisfied with results

RiskPaths.exe -OpenM.TaskName Childlessness -OpenM.TaskWait true

# # find your modeling task run id, i.e.: 1234
# # analyze model output tables
# # if results not optimal
#   # then append new set of input data into task "Childlessness" and continue loop
#   # else signal to RiskPaths model "task can be completed":
#   #   UPDATE task_run_lst SET status = 'p' WHERE task_run_id = 1234;
#
# Done.
#
```

Cluster run: model run on multiple computers



You use MPI to run the model on multiple computers over network or in cloud or on HPC cluster. For example, to run 4 instances of modelOne.exe with 2 threads each and compute 16 sub-values:

```
mpiexec -n 4 modelOne.exe -OpenM.Threads 2 -OpenM.SubValues 16
```

Please notice, usage of `"mpiexec -n 4 ...."` as above is suitable for test only and you should use your cluster tools for real model run.

Cluster task: run modeling task on multiple computers

Modeling task with 1000x input data sets can take long time to run and it is recommended to use cluster (multiple computers over network) or cloud, such as Google Compute Engine, to do that. For example, RiskPaths task above can be calculated much faster if 200 servers available to run it:

```
mpiexec -n 200 RiskPaths.exe -OpenM.TaskName Childlessness -OpenM.SubValues 16 -OpenM.Threads 4
```

Please notice, usage of `"mpiexec -n 200 ...."` as above is suitable for test only and you should use your cluster tools for real model run.

Dynamic task: you can use `-OpenM.TaskWait true` argument as described above to dynamically change task as it runs.

MIT License, Copyright and Contribution

OpenM++ is a Free and Open Source Software

OpenM++ is a free and open source software, licensed under MIT License.

It is free to use, copy, modify, merge, publish, distribute, sublicense, and/or sell this software, for any purpose, commercial or non-commercial.

All openM++ code has been written from scratch and not been taken from other projects.

To find out more about openM++ license please refer:

- MIT License
- open source licenses
- Wiki article on MIT License

OpenM++ License

Copyright Holders for OpenM++

OpenM++ includes contributions by various people and organizations. All OpenM++ contributors retain copyright on their contributions, but agree to release it under the same license as OpenM++.

Contribute to OpenM++

OpenM++ is currently in an intensive foundational development phase of (mostly) full-time developers, which precludes most independent incremental fixes or enhancements. Nevertheless, if you or your organization would like to join and contribute to OpenM++ in this phase, please contact us at openmpp99@gmail.com.

To contribute to OpenM++, we also require that you send us an email indicating that you accept the Developer’s Certificate of Origin (DCO). Basically, the DCO affirms that you have the right to make contributions under the open source license of OpenM++. For more information on DCO's see Contributor Agreements. Here's the text of the DCO used for OpenM++ (taken from the Linux DCO).

```
Developer Certificate of Origin
Version 1.1

Copyright (C) 2004, 2006 The Linux Foundation and its contributors.
660 York Street, Suite 102,
San Francisco, CA 94110 USA

Everyone is permitted to copy and distribute verbatim copies of this
license document, but changing it is not allowed.


Developer's Certificate of Origin 1.1

By making a contribution to this project, I certify that:

(a) The contribution was created in whole or in part by me and I
    have the right to submit it under the open source license
    indicated in the file; or

(b) The contribution is based upon previous work that, to the best
    of my knowledge, is covered under an appropriate open source
    license and I have the right under that license to submit that
    work with modifications, whether created in whole or in part
    by me, under the same open source license (unless I am
    permitted to submit under a different license), as indicated
    in the file; or

(c) The contribution was provided directly to me by some other
    person who certified (a), (b) or (c) and I have not modified
    it.

(d) I understand and agree that this project and the contribution
    are public and that a record of the contribution (including all
    personal information I submit with it, including my sign-off) is
    maintained indefinitely and may be redistributed consistent with
    this project or the open source license(s) involved.
```

The email must contain the DCO text above and indicate your acceptance. Also include the Source Forge user name you will be using for your contributions to OpenM++. If you are contributing as an employee, use your organizational email address and ensure that your hierarchical supervisor(s) are on the CC. If you are not the IP owner of your contributions, provide the name of the organization which is, e.g. Government of Canada.

Your email will be archived and a copy will be placed in the project repository to document the provenance of the contributions you make using your GitHub user ID. Your name will be added to the AUTHORS.txt file of the project. If applicable, the AUTHORS.txt will also indicate that your organization is a contributor and holds copyright to portions of OpenM++.

Usage of other software in OpenM++

As any other product openM++ is using software libraries licensed under different terms. For example, if you choose to use SQLite as openM++ embedded database then SQLite Public Domain license is applied to SQLite portion of openM++. Or, in case of libiconv library, openM++ is using it under LGPL v3.0 license.

Build Files

Some intermediate development files used as part of openM++ build process also fall under other licenses. For example, Microsoft Visual Studio project files or GNU make files. Nothing from such intermediate files ever reaches the final openM++ deliverable and the licenses associated with those building tools should not be a factor in assessing your rights to copy and use openM++.

Model Code: Programming a model

Home > Model Development Topics > **Model Code**

This topic contains general information about the source code of an OpenM++ model. It describes model source code in broad terms, the contents of the model source code folder, and the Default scenario. It also briefly outlines the build process which transforms model source code and a Default scenario into an executable and accompanying database.

Topic contents

- Coding a model
- Code folder and source files
- Source file content
- Default scenario
- Model build
- Hiding syntactic islands

Modgen-specific: References to Modgen in this documentation refer to the Statistics Canada Modgen↗ platform. In this wiki, a model with common source code from which either a Modgen executable or an OpenM++ executable can be built is called a *cross-compatible model*. Wiki content apecific to existing Modgen users, cross-compatible models, or models originally developed in Modgen is highlighted *Modgen-specific* in the text.

Coding a model

OpenM++ models are written in two languages: the OpenM++ language and the C++ language. The OpenM++ language is used to specify the *declarative* aspects of a model, for example the model's classifications, parameters, entities, attributes, events, tables, labels, and notes. The C++ language is used to specify the *procedural* aspects of a model, for example the sequentially executed statements which change an entity's attributes when an event occurs in the simulation.

The OpenM++ language

The OpenM++ language consists of declarative statements. The location and ordering of those statements in model source code files is arbitrary and has no effect on the model specification. This provides a high level of modularity in model source code which can be particularly useful in large and complex models.

A statement in the OpenM++ language starts with an opening keyword which specifies the nature of the declaration and ends with a closing `;`. The syntax between the opening keyword and the closing `;` depends on the nature of the declaration.

For example, the `classification` keyword is used to declare a named ordered list of symbolic values:

```
classification SEX //EN Sex
{
  //EN Male
  MALE,

  //EN Female
  FEMALE
};
```

This example declares an OpenM++ `classification` named `SEX`. It has two possible values `MALE` and `FEMALE`. The declaration of `SEX` means that `SEX` can be used as the dimension of a parameter or table, or as the type (characteristic) of an attribute of an entity in the simulation.

The OpenM++ language also recognizes specially formatted `//` and `/* ... */` comments. Recognized comments are optional and do not affect the model specification. They contain textual information stored with the model which can be used to produce more human-readable input and output and a generated user interface for the model. OpenM++ is multilingual, and the human language of the textual information is specified inside the comment using a two-letter code.

The `//EN` comments in the example provide English-language labels for the `SEX` classification and its values. These labels will appear in the user interface of the model, for example as row or column headings and labels of multi-dimensional parameters and tables.

The C++ language in model code

The C++ language portion of model code consists mostly or entirely of C++ function definitions. Here's an example:

```
// The implement function of MortalityEvent
void Person::MortalityEvent()
{
    alive = false;

    // Remove the entity from the simulation.
    Finish();
}
```

This C++ model code defines the function which implements mortality in the simulation. The `Person` entity, its attribute `alive`, its event `MortalityEvent`, and the helper function `Finish` are all declared elsewhere in the OpenM++ language code of the model.

Typically only a small, limited portion of the C++ language is used in model code. Note that it is usually neither useful nor recommended for a model developer to create C++ classes and class hierarchies in model code. The C++ classes and objects required for simulation are pre-generated by OpenM++ from the model specification given in the OpenM++ language.

The C++ language elements most used in model code are expressions↗ to compute values, assignments↗ to store those values, if statements↗ to implement branching logic, and for statements↗ or range for↗ statements for iteration. C++ functions↗ are used to specify when events occur and what happens when they do. Functions are also used to compute derived parameters and derived tables. Functions can also be used facultatively to organize code in complex models.

The C++ standard library can be used in model code. It includes useful and powerful components such as array↗ and vector↗ in the containers↗ library, and supports string operations.

The limited dialect of C++ used for coding models can be explored by perusing the source code of existing models and referring to comprehensive C++ documentation↗ when necessary, or to the many C++ tutorials available on the web.

Modgen-specific: Unlike Modgen, OpenM++ does not modify the C++ language portions of model code. This provides logical clarity and allows an IDE and other tools to function correctly with the C++ code of a model.

Model symbols in OpenM++ and C++

Many of the named symbols declared in the OpenM++ code of a model are transformed by OpenM++ into identically named C++ symbols for use in the C++ code of the model. The `alive` attribute of the `Person` entity in the previous example is such a symbol. These C++ symbols can usually be used transparently in C++ model code even though they may be implemented as more complex C++ objects 'under the hood'. So, when `alive` is assigned the value `false` in the example, the C++ symbol `alive` will silently implement side-effects to update any tables, derived attributes, or events which depend on the change in its value. Incidentally, these wrapped objects have no memory overhead (the `alive` attribute consumes a single byte of memory) and little computational overhead.

There are some situations where the objects which implement entity attributes can produce unexpected C++ compiler error messages in C++ model code. For more on this issue and how to address it, see Entity attributes in C++.

Model functions in OpenM++ and C++

OpenM++ ignores function definitions in the C++ language portions of model code, with several exceptions:

- Event time function definitions in model code are parsed by OpenM++ to determine which attributes can affect the event time. An event time function will be called to recompute the event time if any of those attributes change value.
- `PreSimulation` function definitions are recognized by OpenM++ and will be called before the simulation starts. `PreSimulation` functions are used to validate input parameters and assign values to derived parameters.
- `UserTables` function definitions are recognized by OpenM++ and will be called after the simulation completes. `UserTables` functions are used to compute the values of derived tables.

[back to topic contents]

Code folder and source files

The source code of an OpenM++ model is in one or more source files (also called modules) located in a single model code folder, eg `Alpha2/code` for the Alpha2 model. Each model source file has a name and extension which determine its language and role when the model is built, as follows:

- `*.h` C++ header files included by other source files.

- `*.cpp` C++ source files, can also contain OpenM++ code **NOT YET IMPLEMENTED**
- `*.mpp` OpenM++ source files, can also contain C++ code
- `*.ompp` OpenM++ source files, can also contain C++ code
- *Modgen-specific*: `modgen_*.mpp` Modgen source files explicitly ignored by OpenM++

Modgen-specific: Only model source files with the .mpp extension are recognized by Modgen. The names and extensions `*.ompp` and `modgen_*.mpp` allow selected model source code files to be processed exclusively by OpenM++ or exclusively by Modgen. This can be useful in cross-compatible models. For example, tables which use the median statistic (which is not supported by Modgen) could be declared in a model source file named `OrdinalStatistics.ompp`. Those tables would be present in the OpenM++ version of the model, but absent in the Modgen version. Declaring those tables in a file with extension `.ompp` means that they will not cause Modgen to stop with a syntax error when building the Modgen version of the model.

The following model-specific source files must be present:

- `custom.h` C++ header file containing model-specific declarations.
- `custom_early.h` C++ header file containing model-specific declarations early in header file inclusion order.

The following model source file is present, by convention:

- `ompp_framework.ompp` Model-specific source file containing `use` statements which specify the names of framework source code modules to be incorporated when the model is built. Framework source code modules are supplied with OpenM++ and are located in the `OM_ROOT/use` folder. For more information, see OpenM++ Framework Library.

Some source files in the OpenM++ model code folder have fixed names and fixed content. Typically a model developer copies them to the model `code` folder from an example model in the OpenM++ distribution, for example from `OM_ROOT/models/NewCaseBased/code` or `OM_ROOT/models/NewTimeBased/code`. They are:

- `case_based.h` Model-independent declaration of a structure present in case-based models, included in `custom.h`.
- *Modgen-specific*: `modgen_case_based.mpp` Model-independent implementation of the simulation core of a case-based Modgen model.
- *Modgen-specific*: `modgen_time_based.mpp` Model-independent implementation of the simulation core of a time-based Modgen model.

[back to topic contents]

Source file content

A model source file can contain only C++ content, only OpenM++ language content, or a mixture of both. OpenM++ uses keywords at the outermost level of code to recognize OpenM++ *syntactic islands* which contain declarative information about the model. Here's an example of an OpenM++ syntactic island in a model source file:

```
parameters
{
    //EN Annual hazard of death
    double MortalityHazard;
    /* NOTE(MortalityHazard, EN)
        A constant hazard of death results in an exponential
        survival function.
    */
};
```

This syntactic island starts with the OpenM++ keyword `parameters` and ends with the terminating `;`.

All code outside of a syntactic island is C++ code. When processing `.mpp` and `.ompp` model code files, OpenM++ extracts all C++ code found outside of syntactic islands and assembles it into the single C++ file `src/om_developer.cpp` for subsequent processing by the C++ compiler. By default, OpenM++ inserts #line directives↗ into this file so that any errors or warnings from the C++ compiler will refer back to the original model source file and line rather than to the assembled file `src/om_developer.cpp`.

When processing a `.cpp` model code file, OpenM++ processes any syntactic islands, but does not extract C++ code outside of syntactic islands. This lets one organize all model code into `.cpp` files in the model code folder, and pass those files directly to the C++ compiler in Step 2 of the model build process (see below). Alternatively one could organize all OpenM++ language content in `.ompp` files, and all C++ language content in `.cpp` files. **NOT YET IMPLEMENTED**

C++ directives can be inserted into model code to improve the usability of an IDE. For more information, see the subtopic Hiding syntactic islands.

Modgen-specific: Modgen processes only `.mpp` files, not `.cpp` files.

[back to topic contents]

Default scenario

The model build process requires a starting scenario containing values for all model input parameters, which is normally named `Default`. The parameter values for the Default scenario are in the model subfolder `parameters/Default`. It is also possible to publish multiple scenarios, not just the Default scenario, when a model is built, see Model Run: How model finds input parameters.

Selected Default parameters can be made invariant and incorporated directly into the model executable. This is done either by placing parameter files into the model subfolder `parameters/Fixed`, or using `parameters_retain` or `parameters_suppress` statements in model code.

The following file types for input parameters are recognized:

- `.dat` Contains values for one or more parameters in Modgen format
- `.odat` Contains values for one or more parameters in Modgen format
- `.csv` Contains values for one parameter in csv format
- `.tsv` Contains values for one parameter in tsv format

Modgen-specific: Only parameter files with the `.dat` extension are recognized by Modgen. The `.odat` extension lets a selected parameter file be processed only by OpenM++. This can be useful in cross-compatible models. It is used in OpenM++ sample cross-compatible models to provide values for parameters which are implemented by scenario properties in Modgen. For example, for the NewCaseBased model, the parameter input file `OM_ROOT/models/NewCaseBased/parameters/Default/Framework.odat` provides values for the `SimulationSeed` and `SimulationCases` parameters. The file `OM_ROOT/models/NewCaseBased/parameters/Default/scenario_info.odat` contains no parameters but provides a label and note for the scenario. Those structured comments would generate an error in Modgen if they were in a `.dat` file.

[back to topic contents]

Model build

The model build process uses the model source code and the Default scenario to construct an executable and accompanying database which implement the model. The model build process can be launched by issuing a command inside an Integrated Development Environment (IDE) such as Visual Studio on Windows, or Visual Studio Code on Linux or MacOS. The build process can also be launched by a command line utility such as `msbuild` on Windows or `make` in Linux. For more information please see Model development in OpenM++. The model build process consists of two steps. Bpth steps can produce warning and error messages. These messages explain the nature of the warning or error and contain the file and line in the model source code. In an IDE, these messages can usually be clicked to navigate directly to the error or wanring location in the IDE code editor.

Many aspects of the OpenM++ framework can be adapted or replaced to work differently or to support other environments. It is also possible to publish models to an existing database and to move or copy published models and scenarios from one database to another. For more information, see subtopics at Home.

Step 1: OpenM++ build

OpenM++ reads and parses all files in the model source subfolder `code` and the files for the Default scenario in `parameters\Default` (and possibly in `parameters\Fixed`), checks for errors, and performs the following steps:

- Extracts the C++ portions of model code from all `.mpp` and `.ompp` files and assembles them into a single C++ source file.
- Generates several C++ header files and a C++ source file which implements the model specification.
- Generates a C++ source file which contains the values of invariant parameters.
- Creates a new empty database for the model.
- Publishes the model's metadata to the database, including classifications, parameter properties, table properties, parameter and table hierarchies, labels and notes, etc.
- Publishes the Default scenario to the database, ie values of all modifiable parameters in the Default scenario.

Step 2: C++ build

After Step 1 completes, the C++ compiler is invoked. The input to the C++ compiler consists of all C++ files in the model source code folder

(`*.cpp`, `*.h`), together with the C++ files generated by OpenM++ in Step 1. Additional general purpose code is included from the OpenM++ distribution and from the C++ standard library.

The results of the C++ compilation are linked with standard C++ libraries and an OpenM++ support library to create the model executable. Because OpenM++ integrates with C++, it is possible to link in other components such as a math library, or even a complete additional model, possibly written in a different language like Fortran.

[back to topic contents]

Hiding syntactic islands

Modern IDEs have powerful abilities to parse and navigate C++ code, e.g. context sensitive popup menus which identify all uses of a symbol in a project. However, these abilities require that the project consist of valid C++. OpenM++ syntactic islands are not valid C++, and will cause errors when processed by an IDE (or an extenral tool like doxygen). Syntactic islands can be hidden from a C++ compiler or IDE by using C++ preprocessor conditional inclusion↗ directives. Here's an example showing how the syntactic island in the earlier example can be hidden from the C++ compiler or IDE.

```
#if 0 // Hide from C++ compiler or IDE
parameters
{
    //EN Annual hazard of death
    double MortalityHazard;
    /* NOTE(MortalityHazard, EN)
        A constant hazard of death results in an exponential
        survival function.
    */
};
#endif // Hide from C++ compiler or IDE
```

OpenM++ will still process the syntactic island because it ignores C++ preprocessor directives.

An IDE may display a hidden syntactic island differently as a visual cue that it's an inactive code block, for example by reducing the opacity of characters in the block to make them fade into the background compared to normal characters. That can make it more difficult to read and edit code in syntactic islands.

To change the display of inactive code blocks in Visual Studio 2022, do
Tools > Options > Text Editor > C/C++ > View
and modify the settings in 'Inactive Code' as desired.

C++ code in model code files will not be considered valid by a C++ compiler or IDE if a required master header file is missing. That's because C++ requires that a symbol be declared before being used in code. That requirement can be met by including the master header file at the top of the model code file, as follows:

```
#include "omc/omSimulation.h" // For C++ compiler or IDE
```

Modgen-specific: In cross-compatible models one may want to disable this `#include` when compiling in the Modgen environment, to avoid passing erroneous declarations to the IDE in the Modgen project and getting spurious error messages referring to OpenM++ constructs. Here's how:

```
#if defined(OPENM)
#include "omc/omSimulation.h" // For C++ compiler or IDE
#endif
```

This works because the manifest constant `OPENM` is always defined in the OpenM++ build environment, and is not defined in the Modgen build environment.

[back to topic contents]

Windows: Create and Debug Models

Where is OpenM++

- Download desktop version: openmpp_win_YYYYMMDD.zip binary files and source code
- Documentation: Windows: Quick Start for Developers

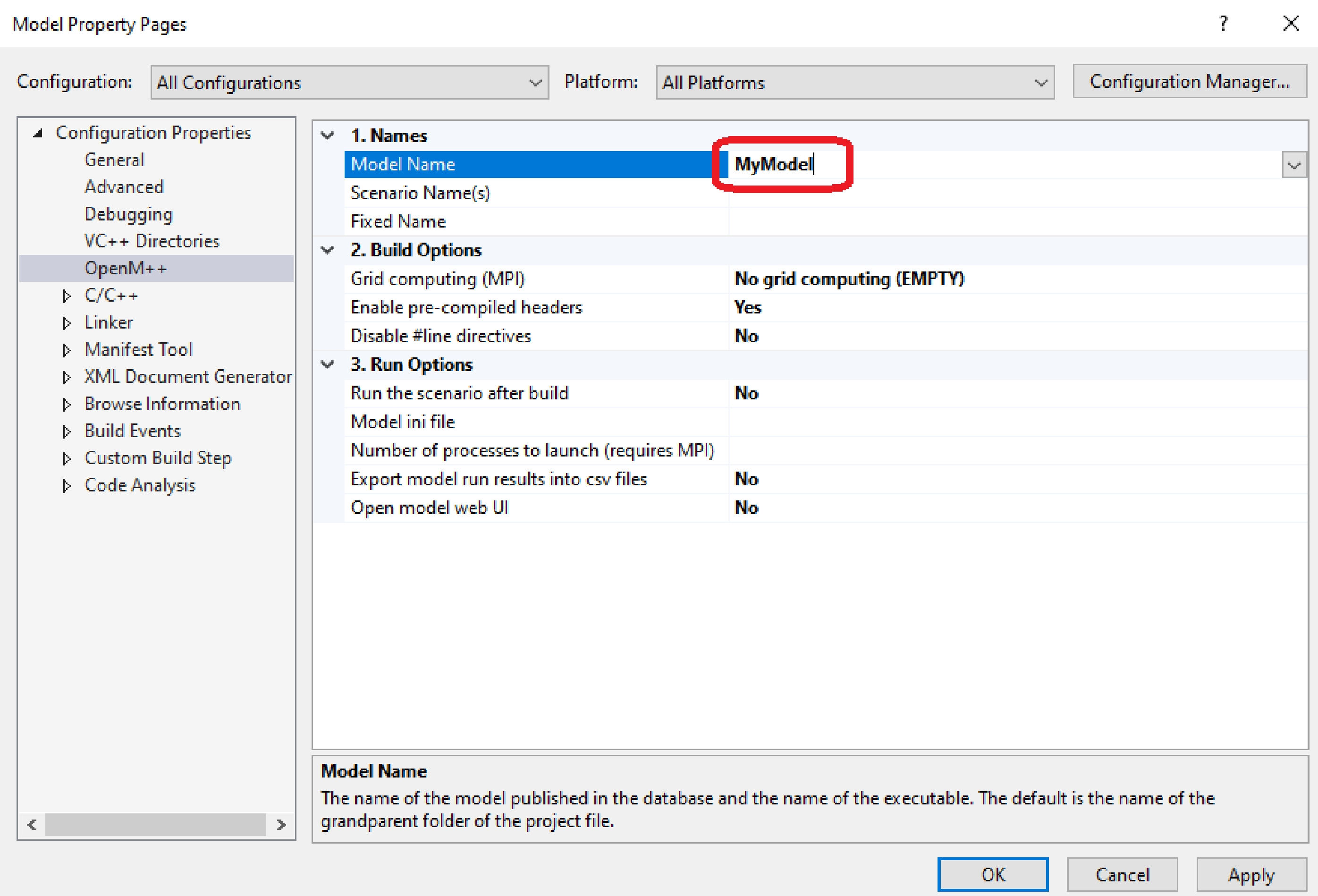
Before you begin

Download and unzip openM++ Windows desktop binaries into any directory, for example: `C:\openmpp_win_20210112\`

Create New Model

- create new directory for your model under models subfolder i.e.: `C:\openmpp_win_20210112\models\MyModel`. It is not required, but recommended to have folder name same as your model name.
- copy one of the test model VC++ project files into your model subfolder, i.e.: from `C:\openmpp_win_20210112\models\NewCaseBased\ompp\*` into `C:\openmpp_win_20210112\models\MyModel\ompp`
- copy your model files `*.ompp *.mpp` and `custom.h` files into `C:\openmpp_win_20210112\models\MyModel\code\` subfolder
- copy your data files `*.odat *.dat` files into `C:\openmpp_win_20210112\models\MyModel\parameters\Default\` subfolder
- start Visual Studio and open `C:\openmpp_win_20210112\models\MyModel\ompp\Model.vcxproj` project
- save your new `Model.sln` solution
- build your model

You can set model name of your new model using Visual Studio menu: Project -> Properties -> Configuration Properties -> OpenM++ -> Name -> Model Name: `MyModel`



Create multiple input sets of parameters (multiple scenarios)

In example above we were creating only one "Default" scenario for our model from *.dat files in `parameters/Default` directory. It is also possible to create multiple input sets of parameters (multiple scenarios) when you are building the model:

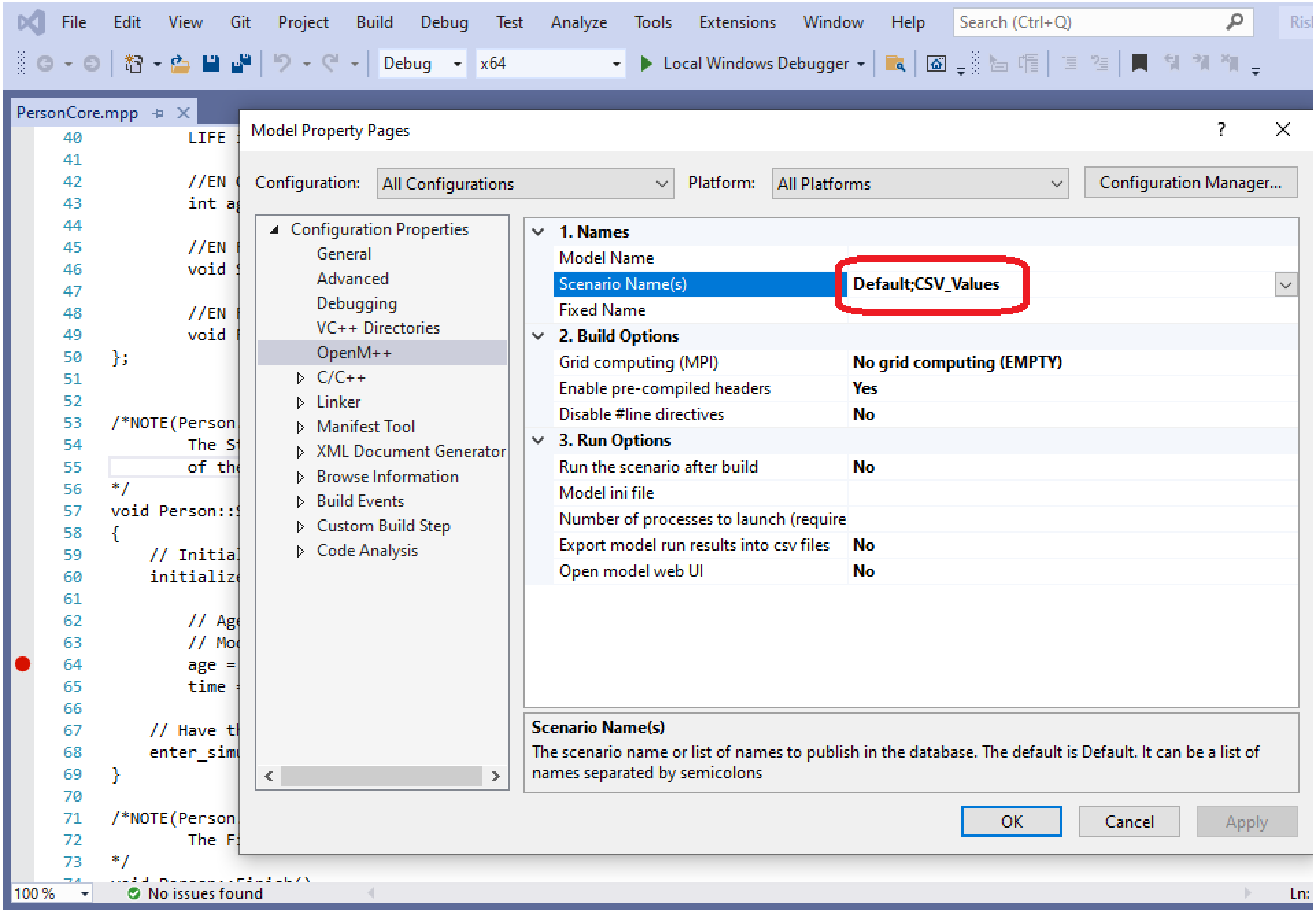
- go to menu: Project -> Properties -> Configuration Properties -> OpenM++
- change: Names -> Scenario Names -> Default;CSV_Values
- Rebuild the project

As result you will create two input sets of parameters in the model.sqlite database:

- scenario "Default" from *.dat, .odat, .csv and* .tsv files in `..\parameters\Default` directory
- scenario "CSV_Values" from *.csv and* .tsv files in `..\parameters\CSV_Values` directory

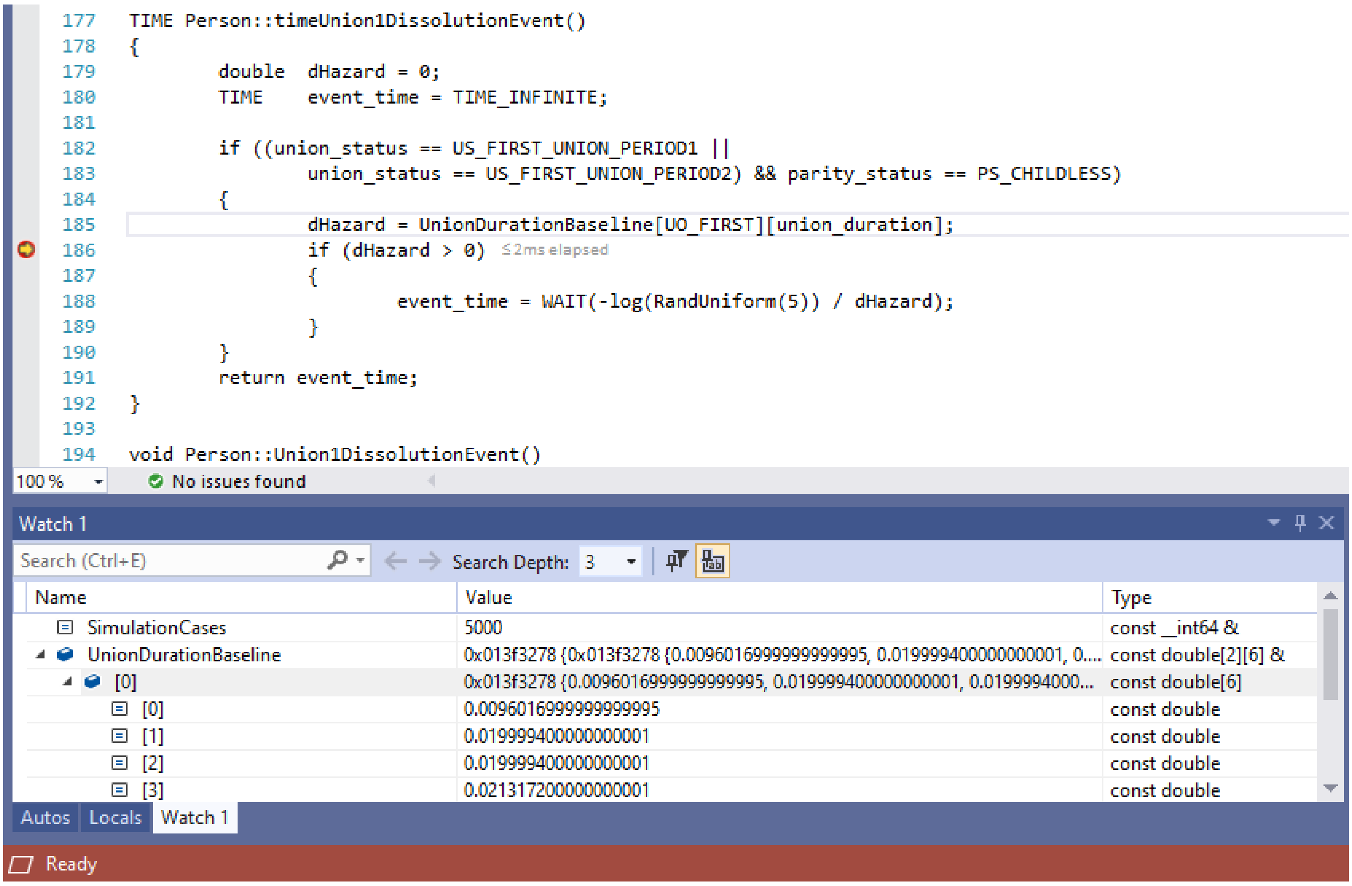
Please notice: additional scenario directory can contain only CSV or TSV files and not .dat or .odat files.

To find out more about CSV and TSV parameter files please read: How to use CSV or TSV files for input parameters values



Debug your Model

- build your model as described above
- open any model.ompp or *.mpp file and put breakpoint in it
- start debugger
- to inspect model parameters go to Watch tab and do "Add item to watch"



Model run options

As described at Windows: Quick Start for Model Users you can run the model with different options. For example, you can calculate 8 sub-values (a.k.a. sub-samples, members, replicas), use 4 threads and simulate 8000 cases:

```
MyModel.exe -OpenM.SubValues 8 -OpenM.Threads 4 -Parameter.SimulationCases 8000
```

You can supply run options as model command line arguments or by using model.ini file:

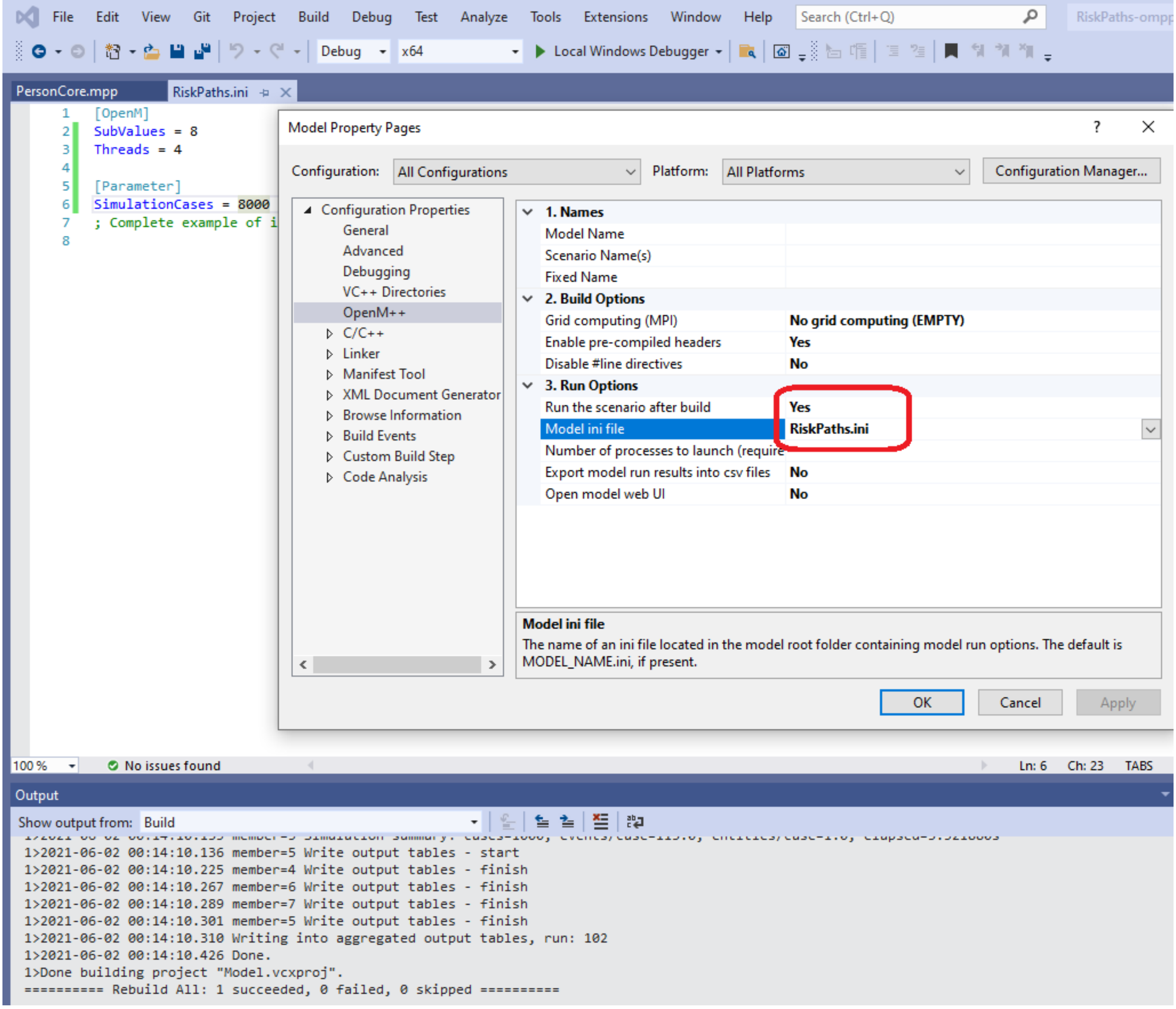
```
[OpenM]
SubValues = 8
Threads = 4

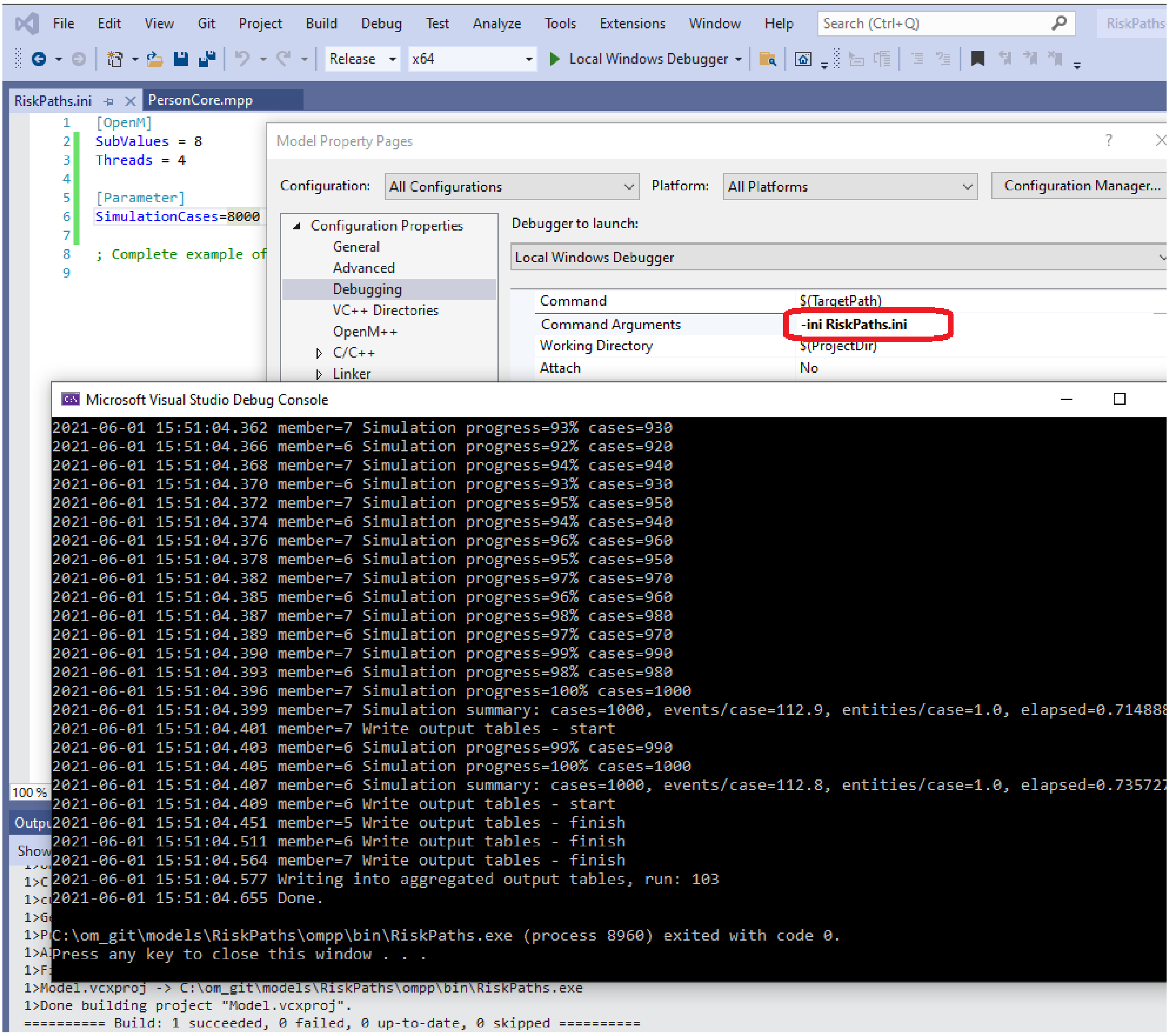
[Parameter]
SimulationCases=8000
```

```
MyModel.exe -ini MyModel.ini
```

There are two possible ways to specify model ini-file using Visual Studio menu:

- Project -> Properties -> Configuration Properties -> OpenM++ -> Run Options
  - Model ini file = `MyModel.ini`
  - Run scenario after build = Yes
- Project -> Properties -> Configuration Properties -> Debugging -> Command Arguments = `-ini MyModel.ini`


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
Debug
x64
Local Windows Debugger
PersonCore.mpp
RiskPaths.ini
[OpenM]
SubValues = 8
Threads = 4
[Parameter]
SimulationCases = 8000
; Complete example of i
Model Property Pages
Configuration: All Configurations
Platform: All Platforms
Configuration Manager...
Configuration Properties
General
Advanced
Debugging
VC++ Directories
OpenM++
C/C++
Linker
Manifest Tool
XML Document Generator
Browse Information
Build Events
Custom Build Step
Code Analysis
1. Names
Model Name
Scenario Name(s)
Fixed Name
2. Build Options
Grid computing (MPI) No grid computing (EMPTY)
Enable pre-compiled headers Yes
Disable #line directives No
3. Run Options
Run the scenario after build Yes
Model ini file RiskPaths.ini
Number of processes to launch (require
Export model run results into csv files No
Open model web UI No
Model ini file
The name of an ini file located in the model root folder containing model run options. The default is MODEL_NAME.ini, if present.
OK
Cancel
Apply
100 %
No issues found
Ln: 6 Ch: 23 TABS
Output
Show output from: Build
1>2021-06-02 00:14:10.136 member=5 Write output tables - start
1>2021-06-02 00:14:10.225 member=4 Write output tables - finish
1>2021-06-02 00:14:10.267 member=6 Write output tables - finish
1>2021-06-02 00:14:10.289 member=7 Write output tables - finish
1>2021-06-02 00:14:10.301 member=5 Write output tables - finish
1>2021-06-02 00:14:10.310 Writing into aggregated output tables, run: 102
1>2021-06-02 00:14:10.426 Done.
1>Done building project "Model.vcxproj".
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========

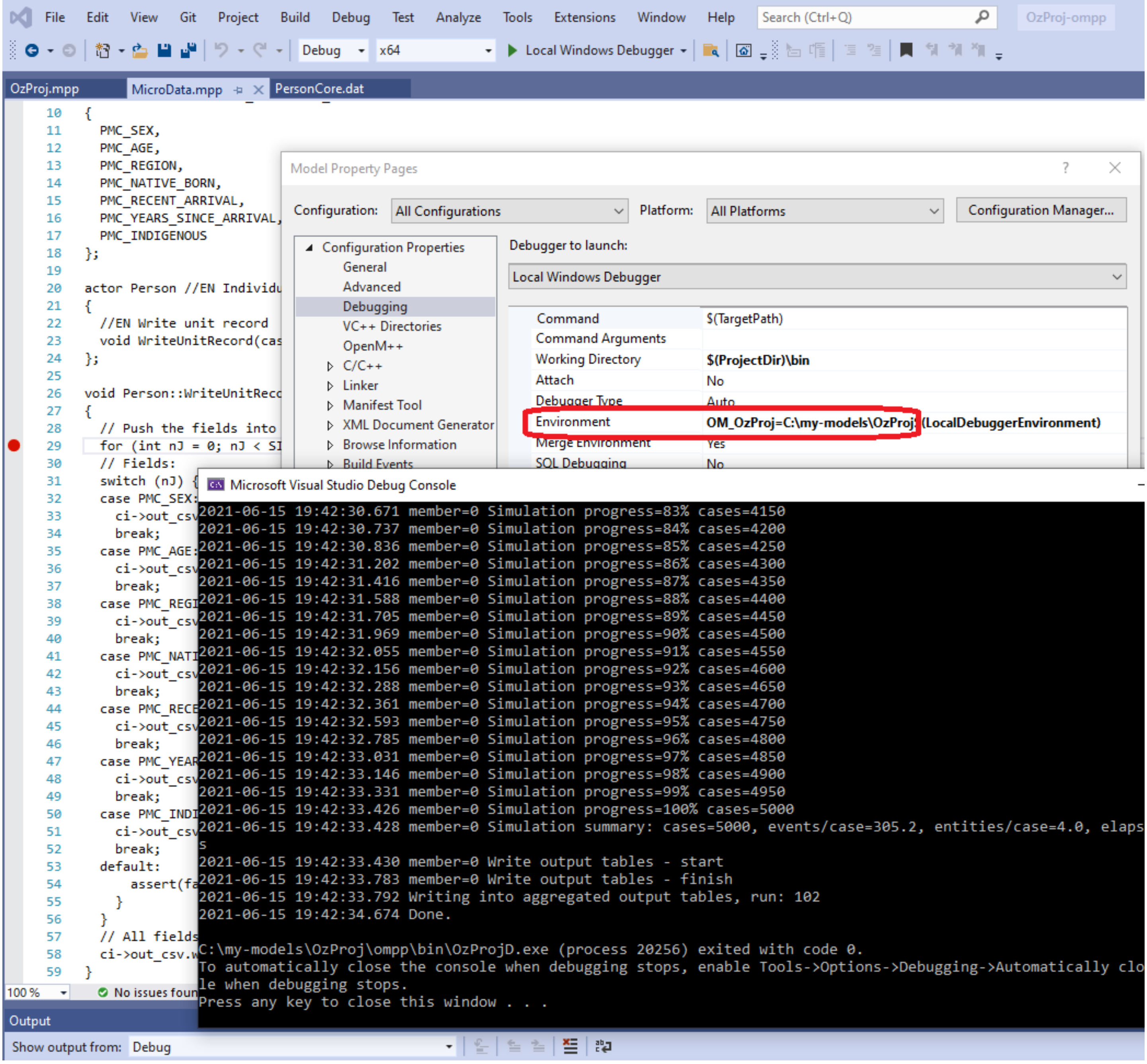


Debug Model with microdata files

If your `BestModel` is using microdata file(s) then it is possible to start microdata path with environment variable:

```
input_csv in_csv;
in_csv.open("$OM_BestModel/microdata/OzProj22_5K.csv");
............
```

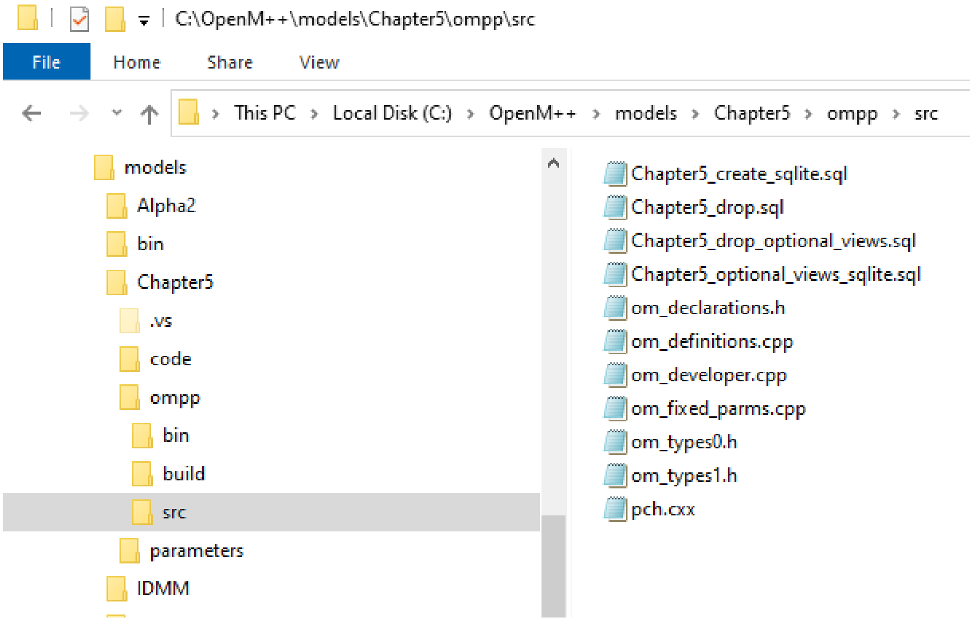
You may need to export that `OM_BestModel` variable in order to debug the model under Visual Studio. For example, if your model location is: `C:\my-models\BestModel` then add: `OM_BestModel=C:\my-models\BestModel` into the model Debugging Environment:



Debug model c++ code

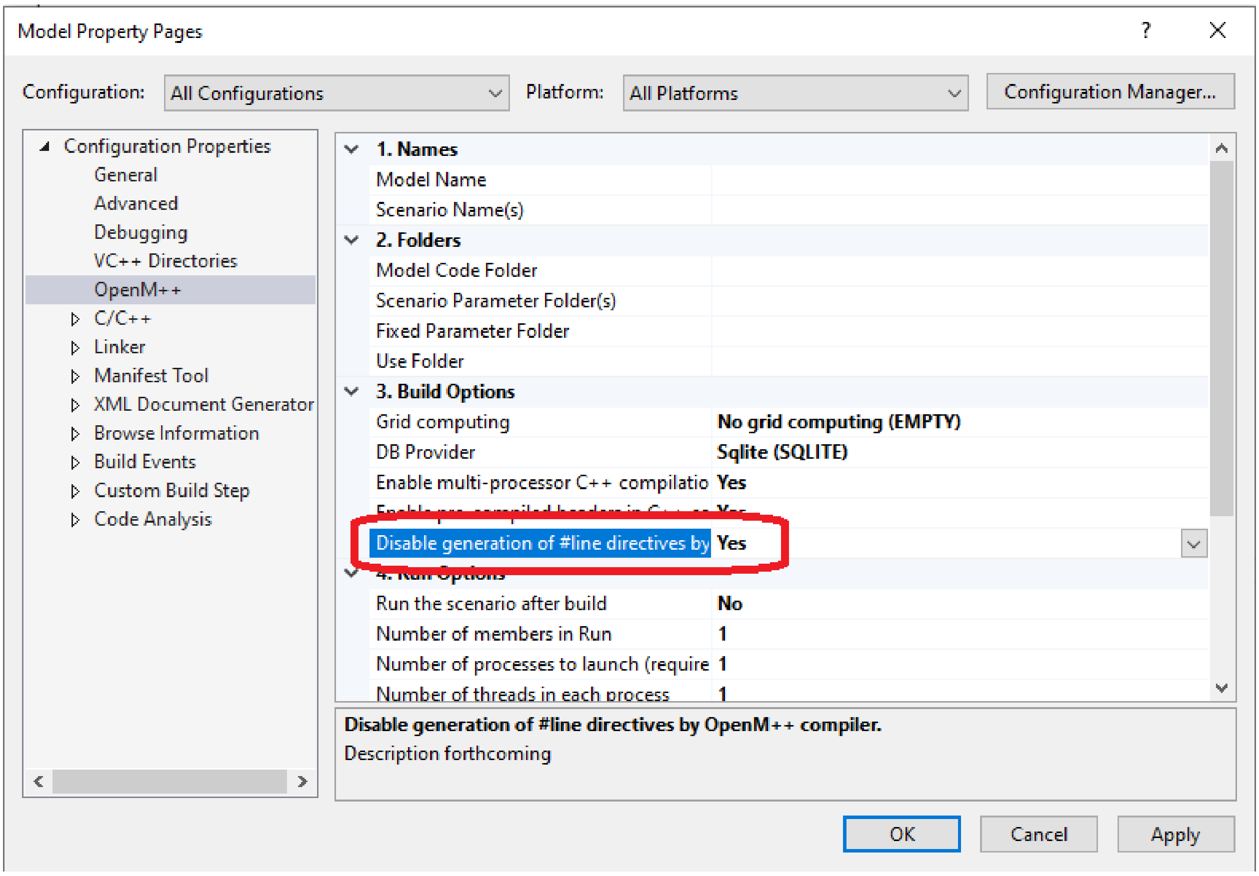
By default model compiled to debug only `*.ompp` and `*.mpp` source code, not a model C++ code. Normally it is enough to debug only `*.ompp` and `*.mpp` code but in some exceptional circumstances you may also want to debug model c++ code, generated by openM++ omc compiler.

C++ model files are located in `ompp/src` directory, for example, if you have openM++ installed in `C:\openmpp_win_20210112` directory then model `Chapter5` *.cpp and* .h source files are in `C:\openmpp_win_20210112\models\Chapter5\ompp\src` folder:

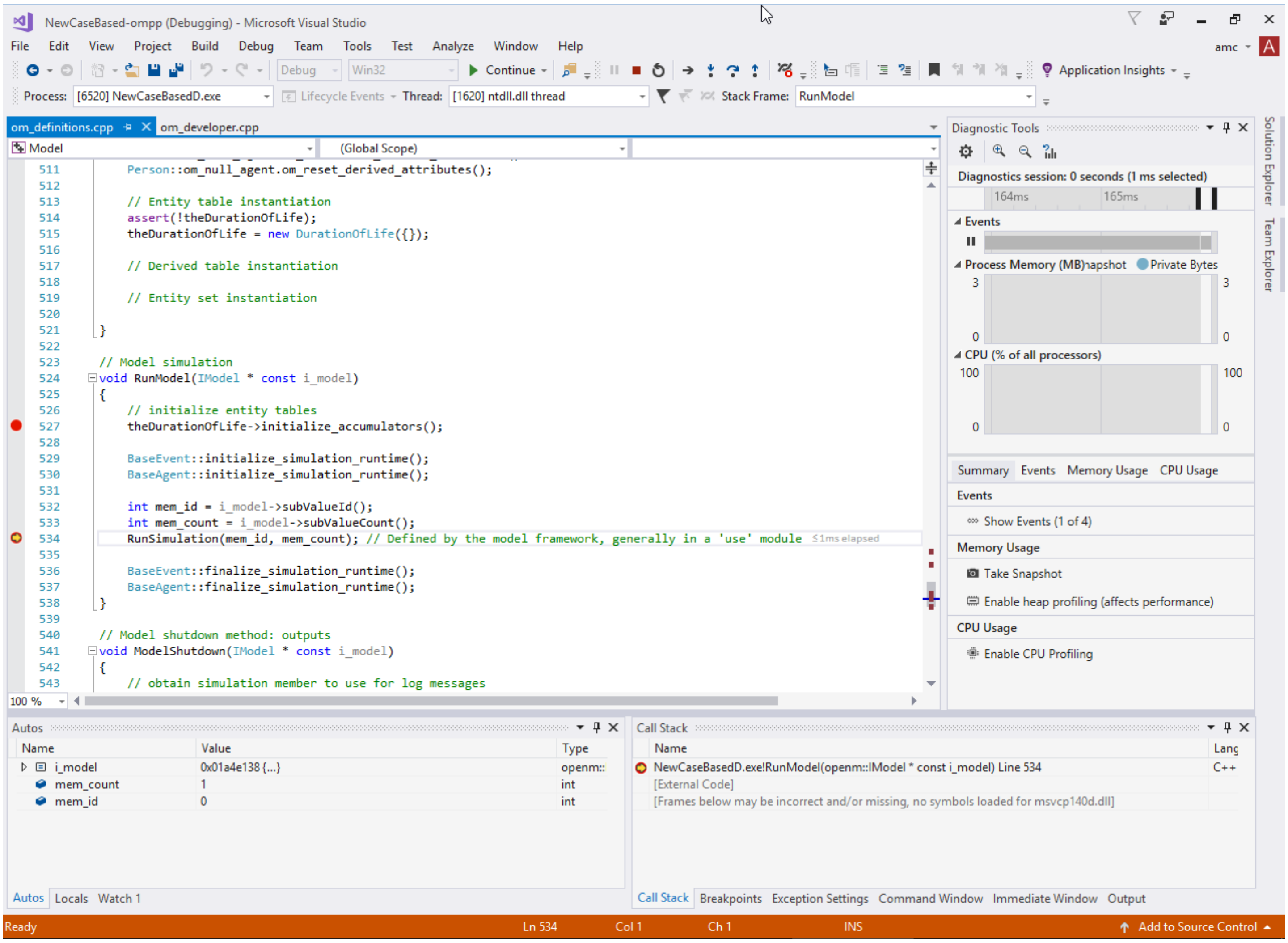


In order to debug model c++ code do following:

- go to menu: Project -> Properties -> Configuration Properties -> OpenM++ -> Disable generation of #line directives = Yes



- Rebuild the model project by going to menu Build -> Rebuild Solution
- put debug breakpoints at the `om_developer.cpp RunSimulation()` or other entry points of your choice, e.g.: `om_definitions.cpp RunModel()`
- start debugger

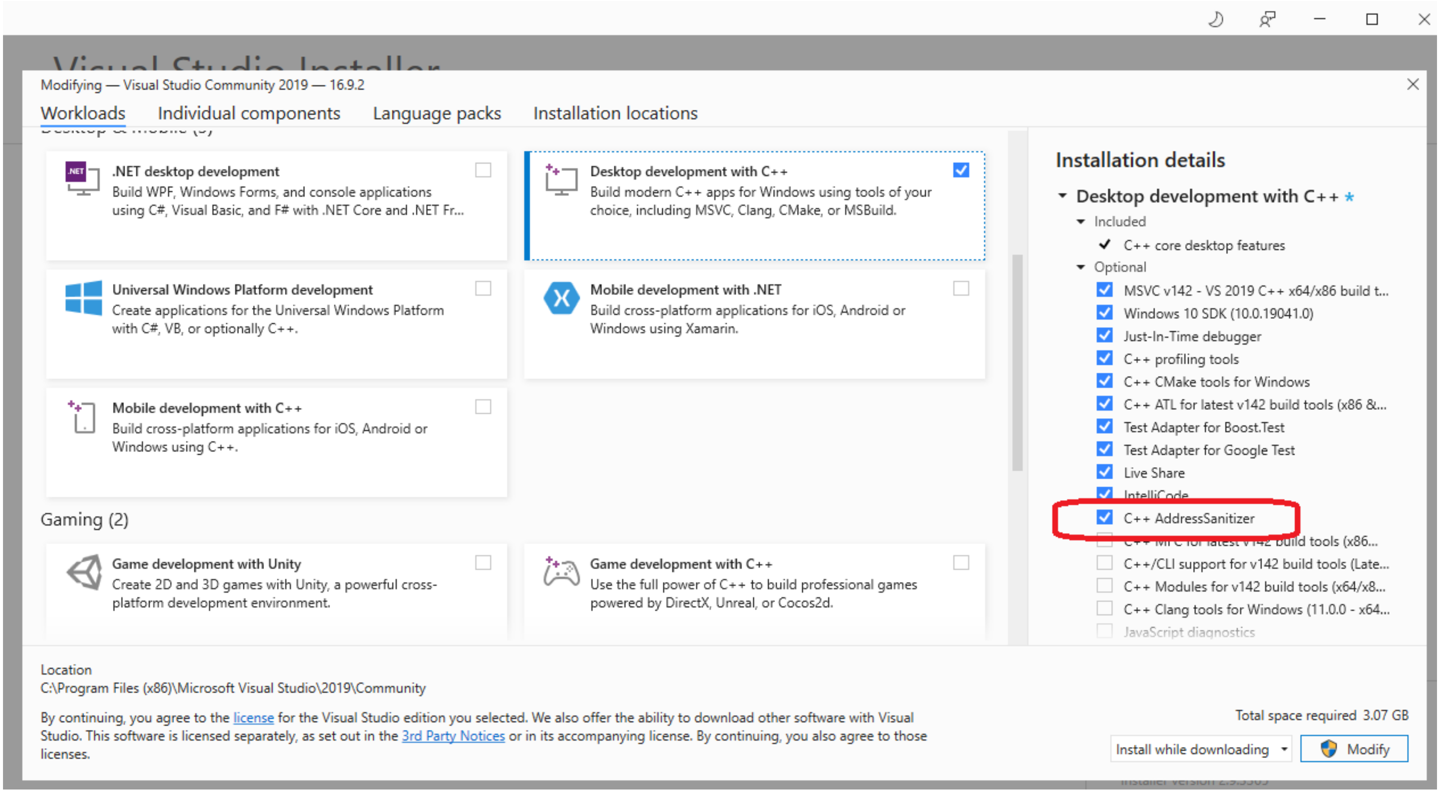


Use AddressSanitizer to catch memory violation bugs

Starting from version 16.9 Visual Studio include AddressSanitizer tool which allow to catch most of memory violation bugs. For example:

```
int x[10];
int main (int argc, char ** argv)
{
    x[20] = 20;  // error: global buffer overflow
    ........
}
```

If you want to add AddressSanitizer to your existing pre-version 16.9 Visual Studio installation start Visual Studio Installer, choose **Modify** and select "C++ AddressSanitizer":



To build your model with AddressSanitizer do following:

- exit from Visual Studio
- copy your existing model project to some backup location:

```
copy C:\openmpp_win_20210112\models\MyModel\ompp\Model.vcxproj* C:\my\safe\place\
```

- copy AddressSanitizer version of model project. For example if openM++ installed into `C:\openmpp_win_20210112` directory and your model directory is `MyModel` then do:

```
copy C:\openmpp_win_20210112\props\ompp-asan\Model.vcxproj C:\openmpp_win_20210112\models\MyModel\ompp\
copy C:\openmpp_win_20210112\props\ompp-asan\Model.vcxproj.filters C:\openmpp_win_20210112\models\MyModel\ompp\
```

- start Visual Studio, open your model openM++ solution `C:\openmpp_win_20210112\models\MyModel\MyModel-ompp.sln`
- **Important: clean existing model build.** You can do it by Menu -> Build -> Clean Solution
- build your model

Now you can run your model from Visual Studio as usually, with or without debugger.

To run model.exe with AddressSanitizer from command line (outside of Visual Studio) use VS 2019 Native Tools command prompt:

- open command line prompt
- set 64 or 32 bit environment:
  - "C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Auxiliary\Build\vcvars64.bat"
  - "C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Auxiliary\Build\vcvars32.bat"
- run your model.exe:

```
cd \openmpp_win_20210112\models\MyModel\ompp\bin
MyModel64D.exe
```

Restore your original Model project from `C:\my\safe\place\Model.vcxproj*` after you done with AddressSanitizer.

Linux: Create and Debug Models

What do you need

- Download: latest binary files and source code
- Documentation: Linux Quick Start for Developers

Before you begin

- check your g++ --version:

```
g++ (Debian 8.3.0-6) 8.3.0
g++ (Ubuntu 9.3.0-10ubuntu2) 9.3.0
g++ (GCC) 8.3.1 20191121 (Red Hat 8.3.1-5)
```

Optional:

If you want to debug your model then you will need to rebuild openM++ runtime library first as described at Linux Quick Start for Developers

To build and run **debug version** of the model use desktop (non-MPI) version of openM++:

```
wget https://github.com/openmpp/main/releases/download/v1.8.3/openmpp_debian_20210304.tar.gz
tar xzf openmpp_debian_20210304.tar.gz
cd openmpp_debian_20210304/openm/
make libopenm
```

Create new Model

- create new directory for your model under models subfolder i.e.: `models/MyModel`
- copy other test model makefile into your model folder, copy your model files and data files:

```
cd openmpp_debian_20210304/models/
mkdir MyModel
cd MyModel
cp ../NewCaseBased/makefile .
mkdir code
cp ~/my_model_sources/*mpp code
cp ~/my_model_sources/*.cpp code
cp ~/my_model_sources/*.h code
mkdir -p parameters/Default
cp ~/my_model_data/*dat parameters/Default
```

- build your model and "publish" it:

```
make all publish
```

- run the model:

```
cd ompp-linux/bin
./MyModelD
cd ..
```

Please note: It is recommended (not required) to have directory name exactly the same as model name. Linux file and directory names are case-sensitive and `myModel` is **not** the same as `MyModel`

Create multiple input sets of parameters (multiple scenarios)

In example above we were creating only one "Default" scenario for our model from *.dat files in `parameters/Default` directory. It is also possible to create multiple input sets of parameters (multiple scenarios) when you are building the model:

```
make SCENARIO_NAME=Default,Other OMC_SCENARIO_PARAM_DIR=parameters/Default,parameters/SomeOther all publish
```

Above command will create two input sets of parameters:

- scenario "Default" from *.dat, *.odat, *.csv and *.tsv files in parameters/Default directory

- scenario "Other" from *.csv and* .tsv files in parameters/SomeOther directory

Please notice: additional scenario directory can contain only CSV or TSV files and not .dat or .odat files.

To find out more about CSV and TSV parameter files please read: How to use CSV or TSV files for input parameters values

Use AddressSanitizer to catch memory violation bugs

There is an excellent AddressSanitizer tool which allow to catch most of memory violation bugs. For example:

```
int x[10];
int main (int argc, char ** argv)
{
   x[20] = 20;  // error: global buffer overflow
   ........
}
```

It is not recommended to use AddressSanitizer in production, it slows down model code execution approximately by 70% and double memory usage. For that reason openM++ binary release does not enable AddressSanitizer by default and you will need to re-build openM++ run-time libraries to use it for your models testing.

To enable AddressSanitizer for your developement do:

- unpack openM++ release in separate folder, for example: `~/openmpp-asan`. It is not recommended to use it in your main development folder
- re-build openM++ run-time library: ``` ` ```bash cd ~/openmpp-asan rm -rf lib rm -rf build

cd openm make USE_ASAN=1 libopenm make USE_ASAN=1 RELEASE=1 libopenm

```
* rebuild your model with AddressSanitizer, for example if your model name is `RiskPaths` you can build Debug and Release model versions by:
```bash
cd ~/ompp-main/models/RiskPaths
make clean-all
make USE_ASAN=1 all publish
make USE_ASAN=1 RELEASE=1 all publish
```

- and now you can run Debug or Release version of your model:

  ```
  cd ompp-linux/bin
  ./RiskPathsD
  ./RiskPaths
  ```

  Please notice, Debug version of the model executable is always significantly slower than Release. It is recommended to prepare smaller version of your test scenario to run it with Debug model. Or, maybe adjust some parameters from default scenario, for example:

  ```
  cd ompp-linux/bin
  ./RiskPathsD -Parameter.SimulationCases 1234
  ```

Debug your Model using Visual Studio Code

Prerequisites:

- install Visual Studio Code
- follow steps described above to create new model

Note: In example below we are using RiskPaths demo model, please replace "RiskPaths" with your actual model name.

Make sure you have GDB, g++, make and other build tools installed on your system. For example on Ubuntu:

```
sudo apt install sqlite
sudo apt install g++
sudo apt install make
sudo apt install curl
sudo apt install git
```

For example on RedHat (CentOS):
```

```
dnf install gcc-c++
dnf install make
dnf install sqlite
dnf install gdb
dnf install git
```

Start Visual Studio Code and go to: File -> Open Folder... -> ~/openmpp_debian_20210304/models/RiskPaths

Create build task for your model using menu: Terminal -> Configure Tasks...

```json
{
  "version": "2.0.0",
  "tasks": [
    {
      "label": "build-RiskPaths",
      "type": "shell",
      "command": "make all publish",
      "problemMatcher": "$gcc",
      "group": {
        "kind": "build",
        "isDefault": true
      },
      "dependsOrder": "sequence",
      "dependsOn": [
        "build-libopenm",
        "stop-ui-RiskPaths"
      ]
    },
    {
      "label": "build-RiskPaths-release",
      "type": "shell",
      "command": "make RELEASE=1 all publish",
      "problemMatcher": "$gcc",
      "group": "build",
      "dependsOrder": "sequence",
      "dependsOn": [
        "build-libopenm-release",
        "stop-ui-RiskPaths"
      ]
    },
    {
      "label": "start-ui-RiskPaths",
      "type": "shell",
      "command": "../start-ompp-ui-linux.sh",
      "problemMatcher": []
    },
    {
      "label": "start-ui-RiskPaths-release",
      "type": "shell",
      "command": "RELEASE=1 ../start-ompp-ui-linux.sh",
      "problemMatcher": []
    },
    {
      "label": "stop-ui-RiskPaths",
      "type": "shell",
      "command": "../stop-ompp-ui-linux.sh",
      "problemMatcher": []
    },
    {
      "label": "clean-RiskPaths",
      "type": "shell",
      "command": "make clean-all && make RELEASE=1 clean-all",
      "group": "build",
      "problemMatcher": []
    },
    {
      "label": "build-libopenm",
      "type": "shell",
      "command": "make libopenm",
      "options": {
        "cwd": "../../openm"
      },
      "problemMatcher": "$gcc",
      "group": "build"
    },
    {
      "label": "build-libopenm-release",
      "type": "shell",
      "command": "make RELEASE=1 libopenm",
      "options": {
        "cwd": "../../openm"
      },
      "problemMatcher": "$gcc",
      "group": "build"
    }
  ]
}
```

Some models may require special settings in order to run, for example, you may need to increase `ulimit` resources for OncSimX model:

```
        {
            "label": "start-ui-OncoSimX",
            "type": "shell",
            "command": "ulimit -S -s 65536 && ../start-ompp-ui-linux.sh",
            "problemMatcher": []
        },
        {
            "label": "start-ui-OncoSimX-release",
            "type": "shell",
            "command": "ulimit -S -s 65536 && RELEASE=1 ../start-ompp-ui-linux.sh",
            "problemMatcher": []
        },
```

Create model debug configuration using menu: Debug -> Add Configuration...:

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "debug RiskPaths",
            "type": "cppdbg",
            "request": "launch",
            "program": "${workspaceFolder}/ompp-linux/bin/RiskPathsD",
            "args": [],
            "stopAtEntry": false,
            "cwd": "${workspaceFolder}/ompp-linux/bin",
            "environment": [
              { "name": "OM_RiskPaths", "value": "${workspaceFolder}" }
            ],
            "externalConsole": false,
            "MIMode": "gdb",
            "setupCommands": [
                {
                    "description": "Enable pretty-printing for gdb",
                    "text": "-enable-pretty-printing",
                    "ignoreFailures": true
                }
            ]
        }
    ]
}
```

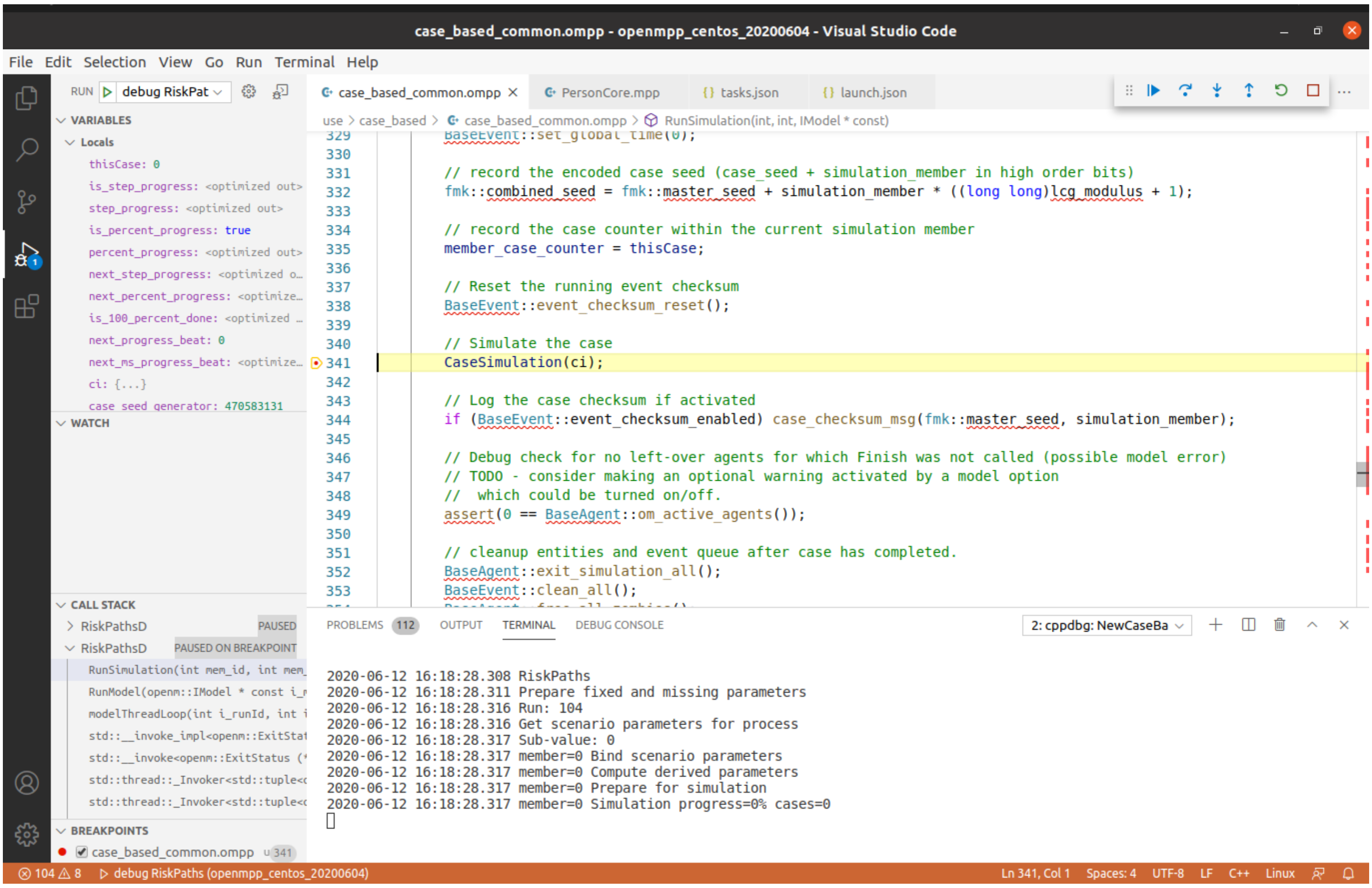
In order to debug *.mpp and* .ompp files as c++ go to menu File -> Preferences -> Settings -> Text Editor -> Files -> Associations -> click on "Edit in settings.json" and add into `settings.json`:

```
{
    "files.associations": {
      "*.mpp": "cpp",
      "*.ompp": "cpp"
    }
}
```

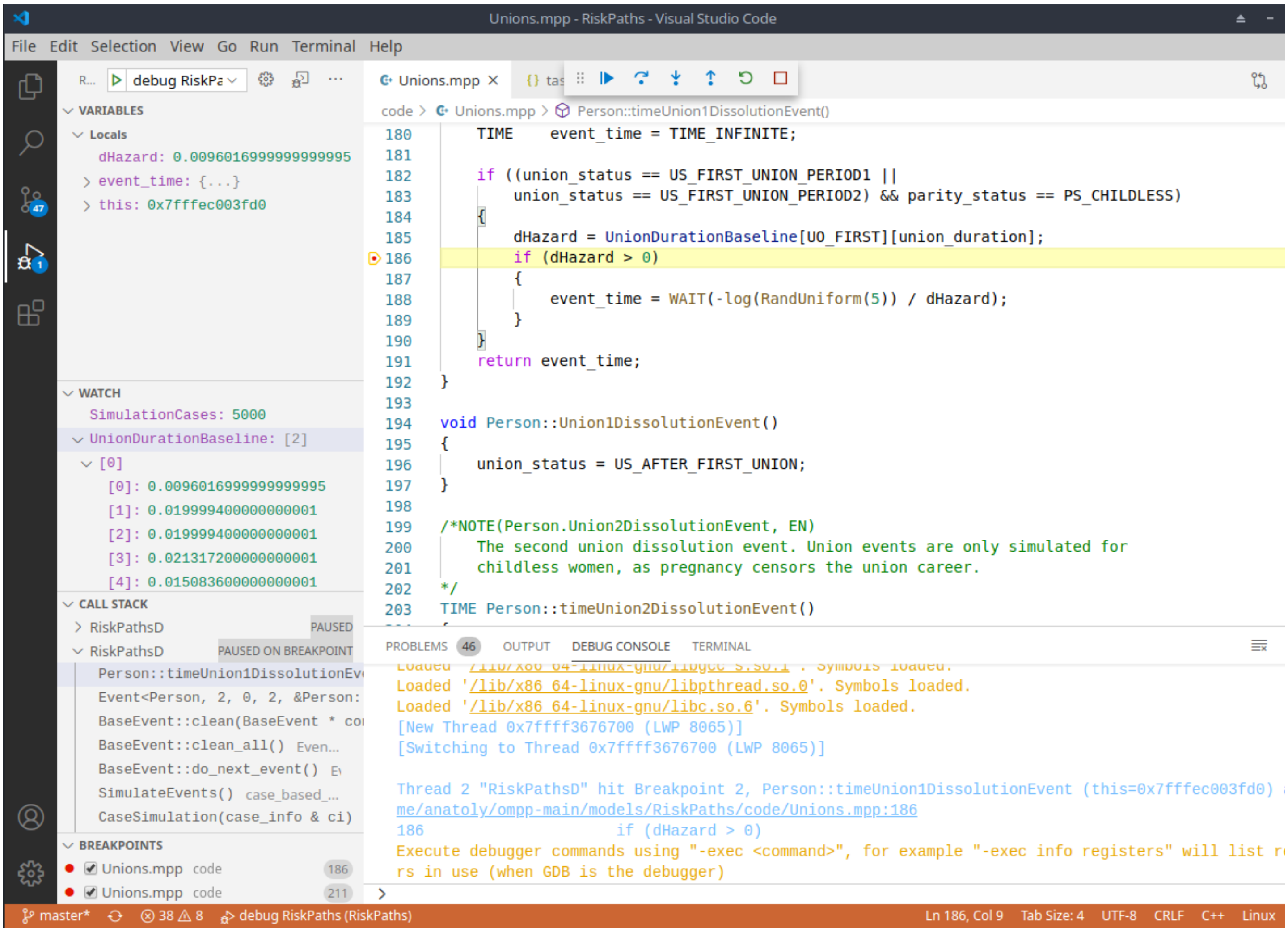
Build your model using Terminal -> Run Build Task...

Start model debugging by using Run -> Start Debugging

- open any model.ompp or *.mpp file and put breakpoint in it
- (optional) add breakpoint(s) at `RunSimulation` entry point using File -> Open File... -> `use/case_based/case_based_common.ompp -> RunSimulation()`
- (optional) you may also add breakpoint(s) at `main` entry point: File -> Open File... -> openm/libopenm/main.cpp
- open model with UI by using Terminal -> Run Task... -> `start-ui-RiskPaths`. You can see UI screenshots at UI: openM++ user interface page.



To inspect model parameters add Watch variable:



Model run options

As described at Linux Quick Start for Model Users you can run the model with different options. For example, you can calculate 8 sub-values (a.k.a. sub-samples, members, replicas), use 4 threads and simulate 8000 cases:

```
./RiskPathsD -OpenM.SubValues 8 -OpenM.Threads 4 -Parameter.SimulationCases 8000
```

You can supply run options as model command line arguments or by using model.ini file:

```
[OpenM]
SubValues = 8
Threads = 4

[Parameter]
SimulationCases=8000
```

```
./RiskPathsD -ini RiskPathsD.ini
```

There are two possible ways to use model ini-file with Visual Studio Code:

- by adding `-ini RiskPaths.ini` command line argument to model executable. Go to menu -> Run -> Open Configurations and edit `launch.json` at `"program"` line:

  ```
  {
   // .... .... ....
   "program": "${workspaceFolder}/ompp-linux/bin/RiskPathsD -ini RiskPaths.ini",
   // .... .... ....
  }
  ```

- by adding `MODEL_INI=RiskPaths.ini` command line argument to model make. Go to menu -> Terminal -> Configure Task -> build-RiskPaths and edit `tasks.json` at `"command": "make ....` line:

  ```
  {
   "tasks": [
       {
           "label": "build-RiskPaths",
           "command": "make MODEL_INI=RiskPaths.ini all publish run",
       // .... .... ....
       }]
  }
  ```

  That `MODEL_INI` argument will be passed to model executable when `make` run the model as:

  ```
  ompp-linux/bin/RiskPathsD -ini RiskPaths.ini
  ```

MacOS: Create and Debug Models

What do you need

- Download: latest binary files and source code
- Documentation:
  - MacOS Quick Start for Developers
  - (optional) MacOS: Create and Debug Models using Xcode

Prerequisites

- Tested on: MacOS 10.15 Catalina And Big Sur >= 11.1.
- Install Xcode and command line developer tools, if not installed already by Xcode: `xcode-select --install`.
- (optional) Install Visual Studio Code for cross-platform development: MacOS: Install VSCode
- Check if clang, make and sqlite3 are installed on your computer:

```
g++ --version
....
Apple clang version 11.0.0 (clang-1100.0.33.12)

make --version
....
GNU Make 3.81

sqlite3 --version
....
3.28.0 2019-04-15 14:49:49
```

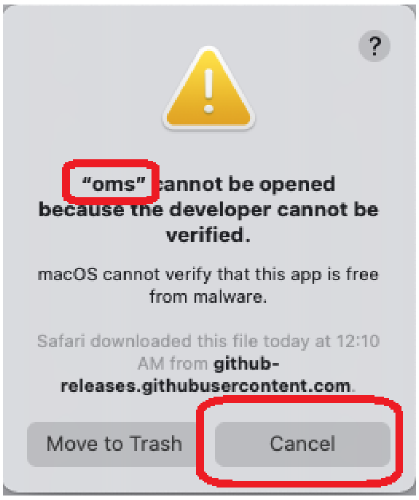
- Download and unpack latest openM++ release using Safari or curl:

```
curl -L -o om.tar.gz https://github.com/openmpp/main/releases/download/v1.6.0/openmpp_mac_20200621.tar.gz
tar xzf om.tar.gz
```

MacOS security issue

Make sure you are using tight security settings on your Mac and antivirus software, if necessary. We are trying our best to keep development machines clean, but cannot provide any guarantee.

On Big Sur it is very likely to get an security error when you are trying to run any downloaded executable:



- please reply "Cancel" to that question (click "Cancel" button).
- remove quarantine attribute from openM++ installation directory, for example:

```
xattr -r -d com.apple.quarantine ~/openmpp_mac_20200621
```

Create new Model

- create new directory for your model under models sub-folder: `models/MyModel` **Please note:** It is recommended (not required) to have directory name exactly the same as model name.
- copy other test model makefile into your model folder, copy your model files and data files:

```
cd openmpp_mac_20200621/models/
mkdir MyModel
cd MyModel
cp ../NewCaseBased/makefile .
mkdir code
cp ~/my_model_sources/*mpp code
cp ~/my_model_sources/*.cpp code
cp ~/my_model_sources/*.h code
mkdir -p parameters/Default
cp ~/my_model_data/*dat parameters/Default
```

- build your model:

```
make all publish
```

- run the model:

```
cd ompp-mac/bin
./MyModelD
cd ..
```

- you can also build and run the model using make:

```
make all publish run
```

Create multiple input sets of parameters (multiple scenarios)

In example above we were creating only one "Default" scenario for our model from *.dat files in `parameters/Default` directory. It is also possible to create multiple input sets of parameters (multiple scenarios) when you are building the model:

```
make SCENARIO_NAME=Default,Other OMC_SCENARIO_PARAM_DIR=parameters/Default,parameters/SomeOther all publish
```

Above command will create two input sets of parameters:

- scenario "Default" from *.dat,* .odat, *.csv and* .tsv files in parameters/Default directory
- scenario "Other" from *.csv and* .tsv files in parameters/SomeOther directory

Please notice: additional scenario directory can contain only CSV or TSV files and not .dat or .odat files.

To find out more about CSV and TSV parameter files please read: How to use CSV or TSV files for input parameters values

Use AddressSanitizer to catch memory violation bugs

There is an excellent AddressSanitizer tool which allow to catch most of memory violation bugs. For example:

```
int x[10];
int main (int argc, char ** argv)
{
    x[20] = 20;  // error: global buffer overflow
    ........
}
```

It is not recommended to use AddressSanitizer in production, it slows down model code execution approximately by 70% and double memory usage. For that reason openM++ binary release does not enable AddressSanitizer by default and you will need to re-build openM++ run-time libraries to use it for your models testing.

To enable AddressSanitizer for your developement do:

- unpack openM++ release in separate folder, for example: `~/openmpp-asan`. It is not recommended to use it in your main development folder
- re-build openM++ run-time library: `` ` ``bash cd ~/openmpp-asan rm -rf lib rm -rf build

cd openm make USE_ASAN=1 libopenm make USE_ASAN=1 RELEASE=1 libopenm

```
* rebuild your model with AddressSanitizer, for example if your model name is `RiskPaths` you can build Debug and Release model versions by:
```bash
cd ~/ompp-main/models/RiskPaths
make clean-all
make USE_ASAN=1 all publish
make USE_ASAN=1 RELEASE=1 all publish
```

- and now you can run Debug or Release version of your model:

  ```
  cd ompp-mac/bin
  ./RiskPathsD
  ./RiskPaths
  ```

  Please notice, Debug version of the model executable is always significantly slower than Release. It is recommended to prepare smaller version of your test scenario to run it with Debug model. Or, maybe adjust some parameters from default scenario, for example:

  ```
  cd ompp-mac/bin
  ./RiskPathsD -Parameter.SimulationCases 1234
  ```

How to use Visual Studio Code

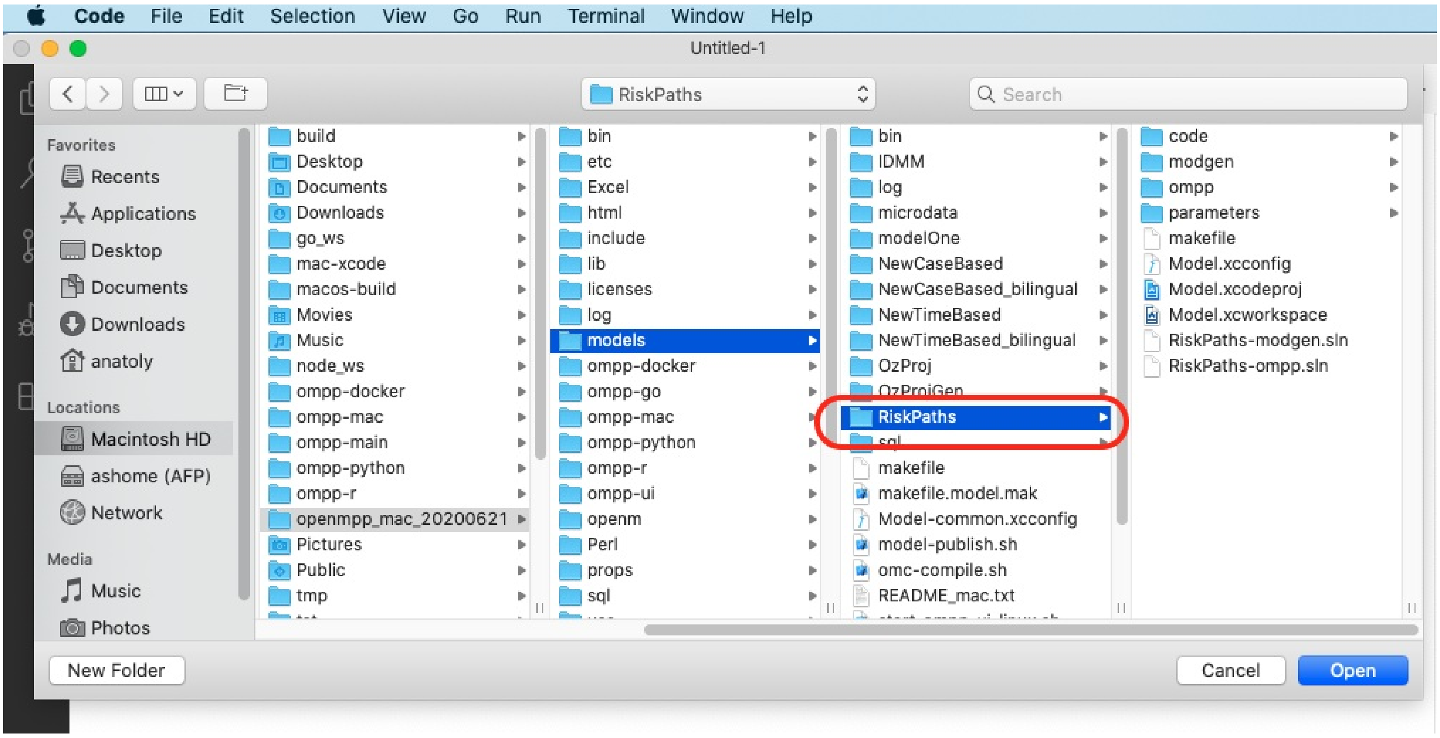
Build openM++ models using VSCode

Prerequisites:

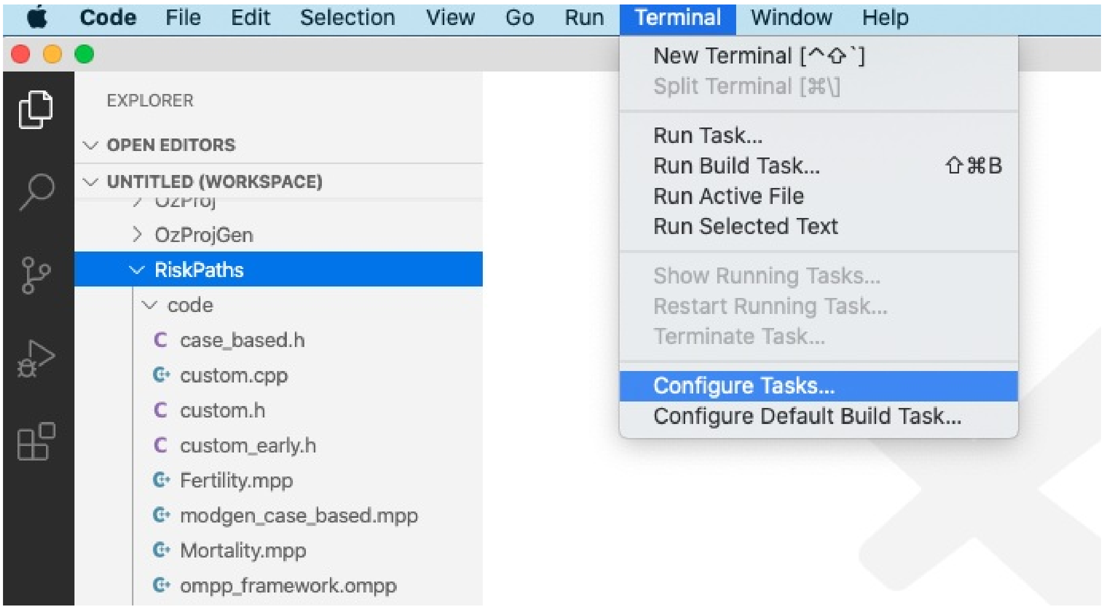
- install Visual Studio Code and configure it for openM++ model development: MacOS: Install VSCode
- follow steps described above to create new model

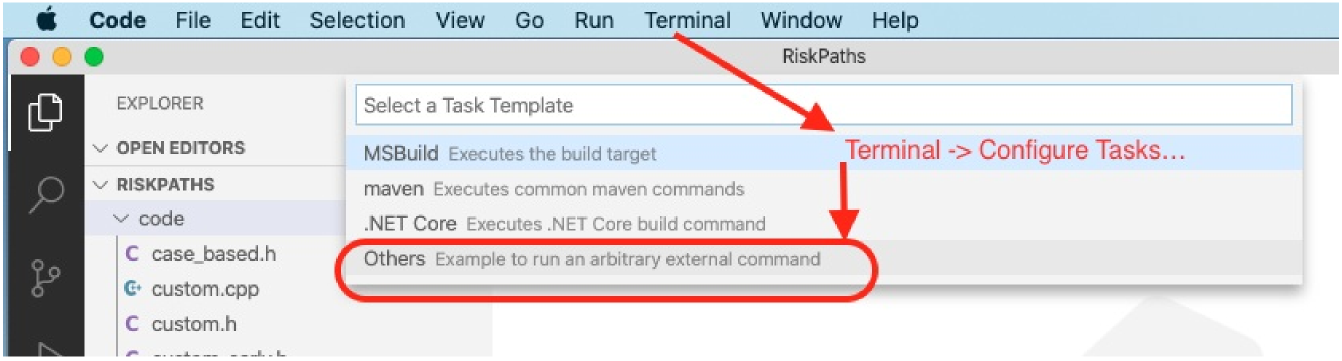
Note: In example below we are using RiskPaths demo model, please replace "RiskPaths" with your actual model name.

Start VSCode and use menu to File -> Open... -> ~/openmpp_mac_20200621/models/RiskPaths:



Configure build tasks by using menu: Terminal -> Configure Tasks...
  ```


Code
File
Edit
Selection
View
Go
Run
Terminal
Window
Help
EXPLORER
OPEN EDITORS
UNTITLED (WORKSPACE)
OzProjGen
RiskPaths
code
case_based.h
custom.cpp
custom.h
custom_early.h
Fertility.mpp
modgen_case_based.mpp
Mortality.mpp
ompp_framework.ompp
New Terminal [^⇧`]
Split Terminal [⌘\]
Run Task...
Run Build Task... ⇧⌘B
Run Active File
Run Selected Text
Show Running Tasks...
Restart Running Task...
Terminate Task...
Configure Tasks...
Configure Default Build Task...


Code
File
Edit
Selection
View
Go
Run
Terminal
Window
Help
RiskPaths
EXPLORER
OPEN EDITORS
RISKPATHS
code
case_based.h
custom.cpp
custom.h
Select a Task Template
MSBuild Executes the build target
maven Executes common maven commands
.NET Core Executes .NET Core build command
Others Example to run an arbitrary external command
Terminal -> Configure Tasks...

```
{
  // See https://go.microsoft.com/fwlink/?LinkId=733558
  // for the documentation about the tasks.json format
  "version": "2.0.0",
  "tasks": [
    {
      "label": "build-RiskPaths",
      "type": "shell",
      "command": "make all publish",
      "problemMatcher": "$gcc",
      "group": {
        "kind": "build",
        "isDefault": true
      },
      "dependsOrder": "sequence",
      "dependsOn": [
        "build-libopenm",
        "stop-ui-RiskPaths"
      ]
    },
    {
      "label": "build-RiskPaths-release",
      "type": "shell",
      "command": "make RELEASE=1 all publish",
      "problemMatcher": "$gcc",
      "group": "build",
      "dependsOrder": "sequence",
      "dependsOn": [
        "build-libopenm-release",
        "stop-ui-RiskPaths"
      ]
    },
    {
      "label": "start-ui-RiskPaths",
      "type": "shell",
      "command": "../start-ompp-ui-mac.sh",
      "problemMatcher": []
    },
    {
      "label": "start-ui-RiskPaths-release",
      "type": "shell",
      "command": "RELEASE=1 ../start-ompp-ui-mac.sh",
      "problemMatcher": []
    },
    {
      "label": "stop-ui-RiskPaths",
      "type": "shell",
      "command": "../stop-ompp-ui-mac.sh",
      "problemMatcher": []
    },
    {
      "label": "clean-RiskPaths",
      "type": "shell",
      "command": "make clean-all && make RELEASE=1 clean-all",
      "group": "build",
      "problemMatcher": []
    },
    {
      "label": "build-libopenm",
      "type": "shell",
      "command": "make libopenm",
      "options": {
        "cwd": "../../openm"
      },
      "problemMatcher": "$gcc",
      "group": "build"
    },
    {
      "label": "build-libopenm-release",
      "type": "shell",
      "command": "make RELEASE=1 libopenm",
      "options": {
        "cwd": "../../openm"
      },
      "problemMatcher": "$gcc",
      "group": "build"
    }
  ]
}
```

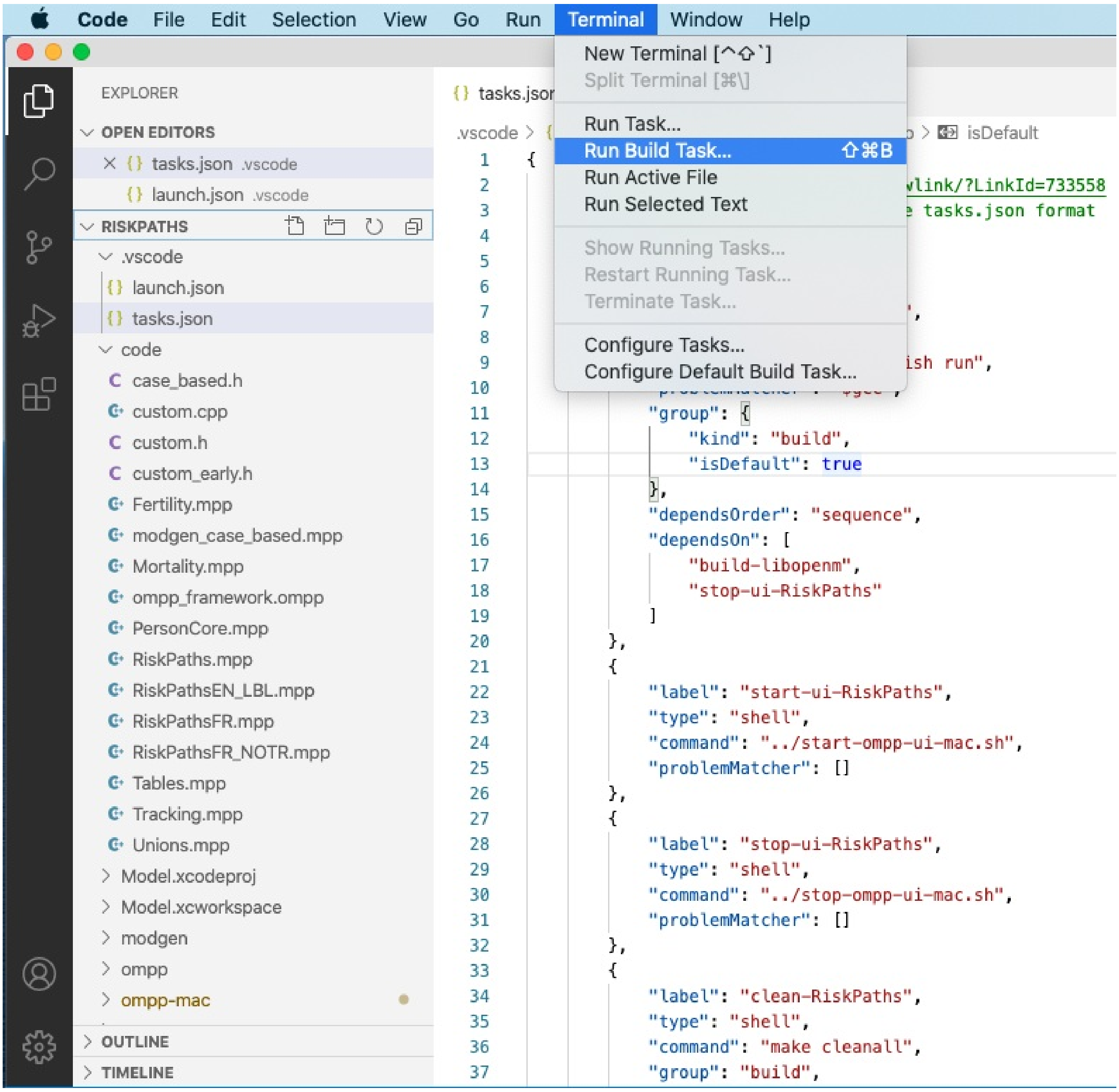
Note: Model default build task `make all publish run` does:

- create Debug version of model executable

- copy model SQLite database file into `ompp-mac/bin` "publish" folder

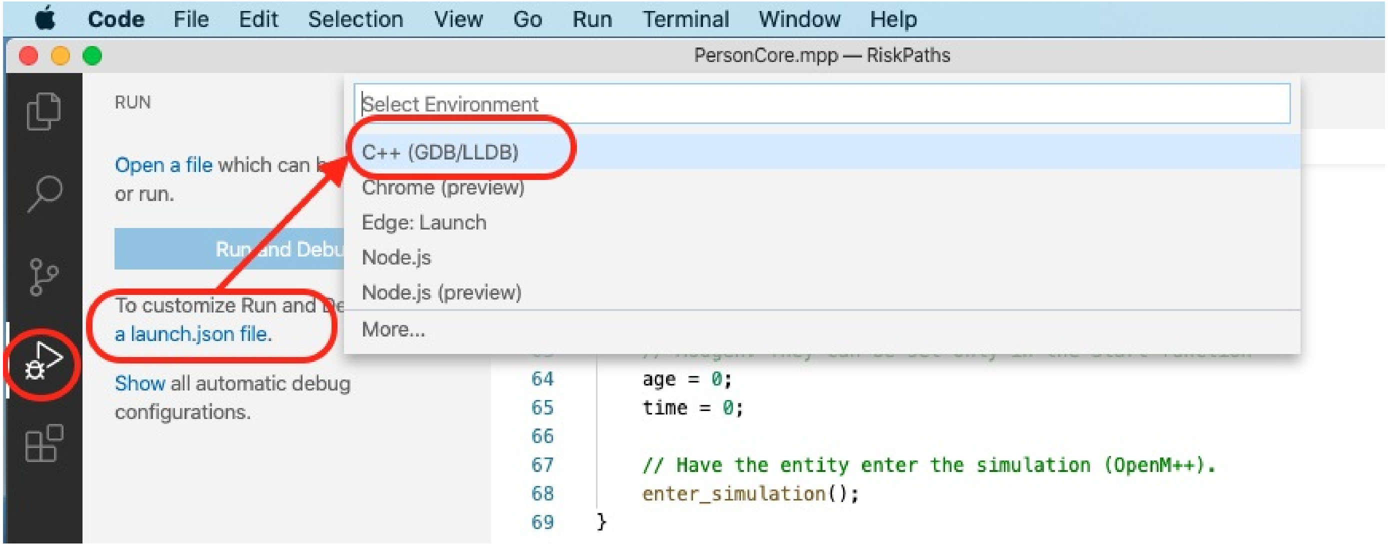
If you also want to run the model after successful build then use: `make all publish run`. If you want to build Release version of the model then use: `make RELEASE=1 all publish`.

To build and run your model please use menu: Terminal -> Run Build Task...



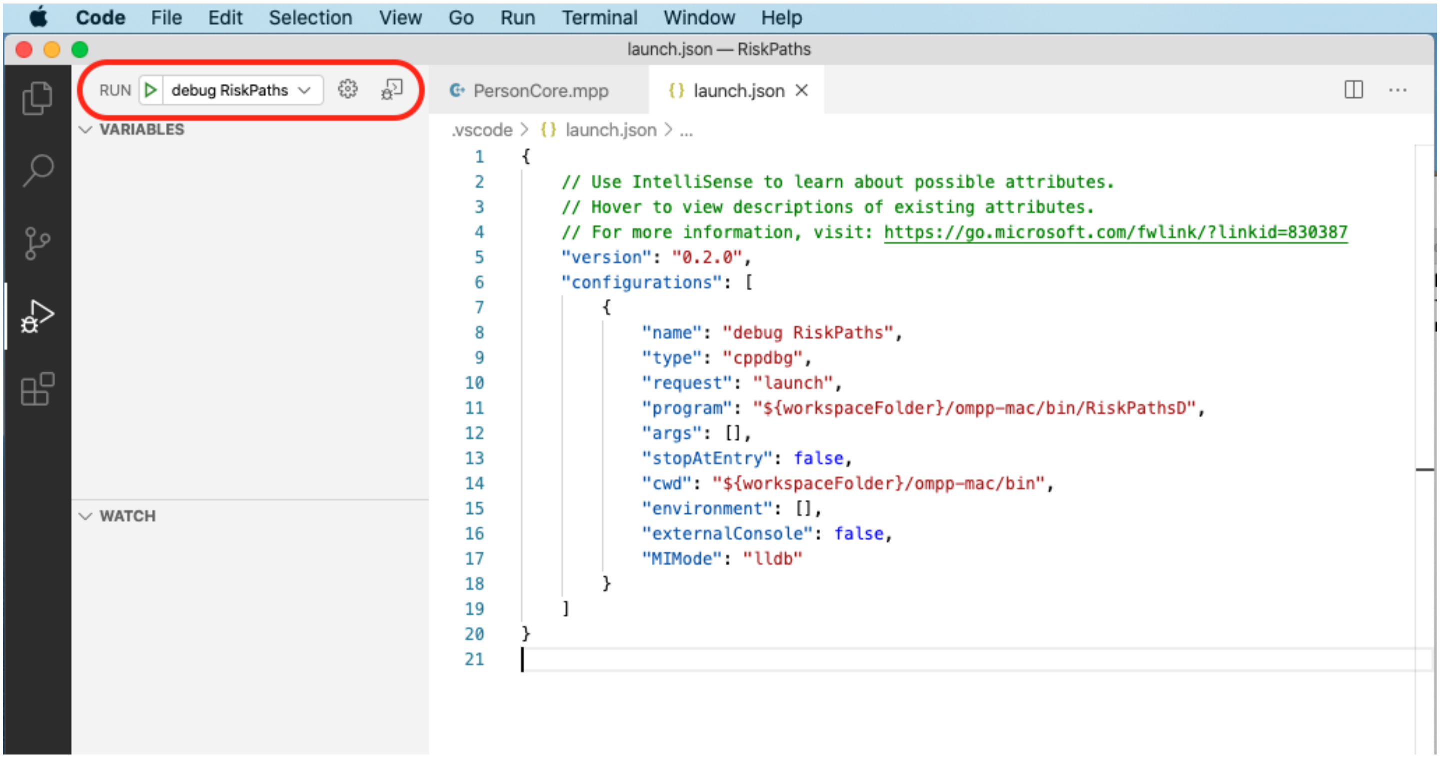
Debug openM++ model using VSCode

Create your model debug configuration by using menu Run -> Add Configuration...




```
{
  // Use IntelliSense to learn about possible attributes.
  // Hover to view descriptions of existing attributes.
  // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
  "version": "0.2.0",
  "configurations": [
    {
      "name": "debug RiskPaths",
      "type": "cppdbg",
      "request": "launch",
      "program": "${workspaceFolder}/ompp-mac/bin/RiskPathsD",
      "args": [],
      "stopAtEntry": false,
      "cwd": "${workspaceFolder}/ompp-mac/bin",
      "environment": [
       { "name": "OM_RiskPaths", "value": "${workspaceFolder}" }
      ],
      "externalConsole": false,
      "MIMode": "lldb"
    }
  ]
}
```

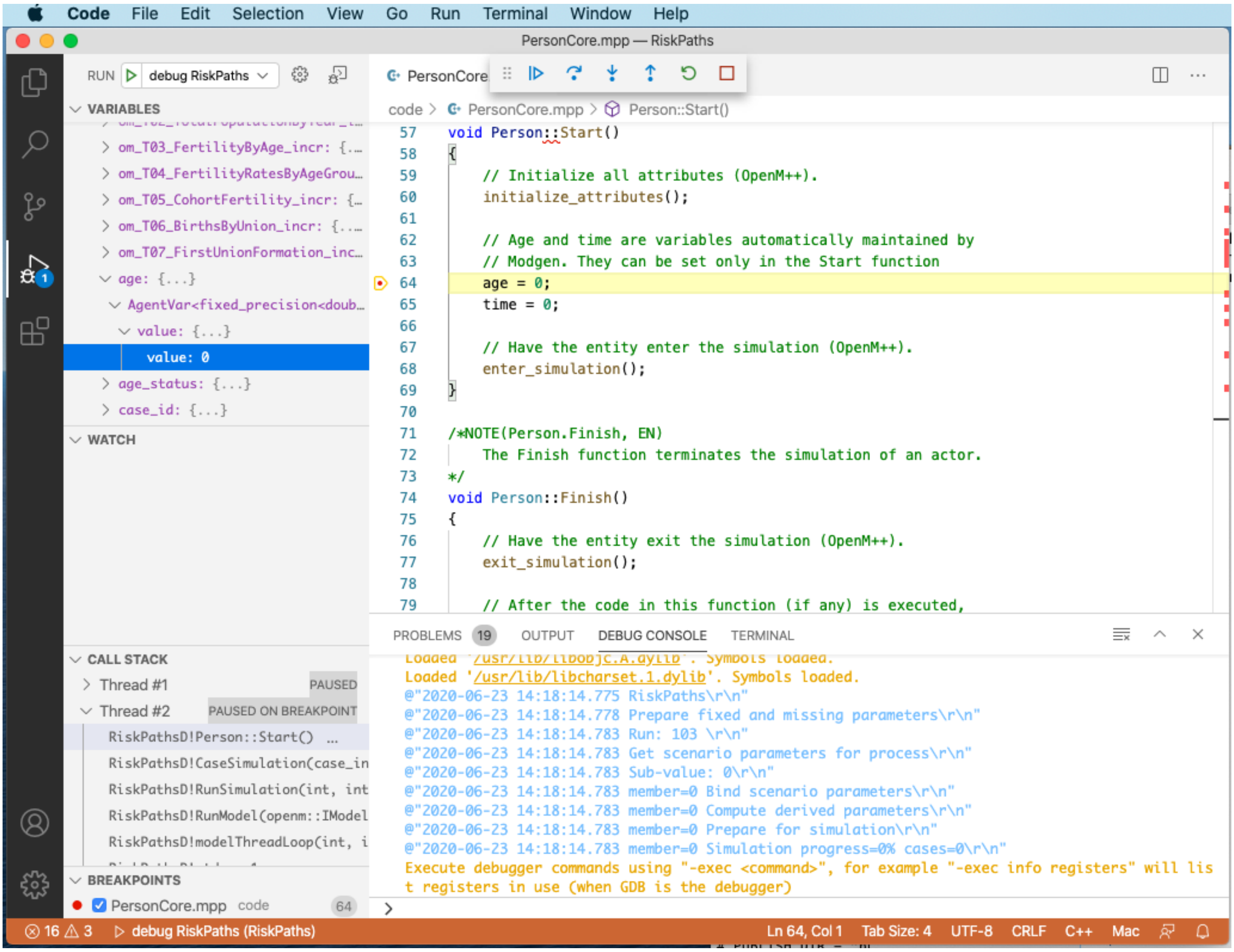
Start model debugging by using menu Run -> Start Debugging or as shown below:



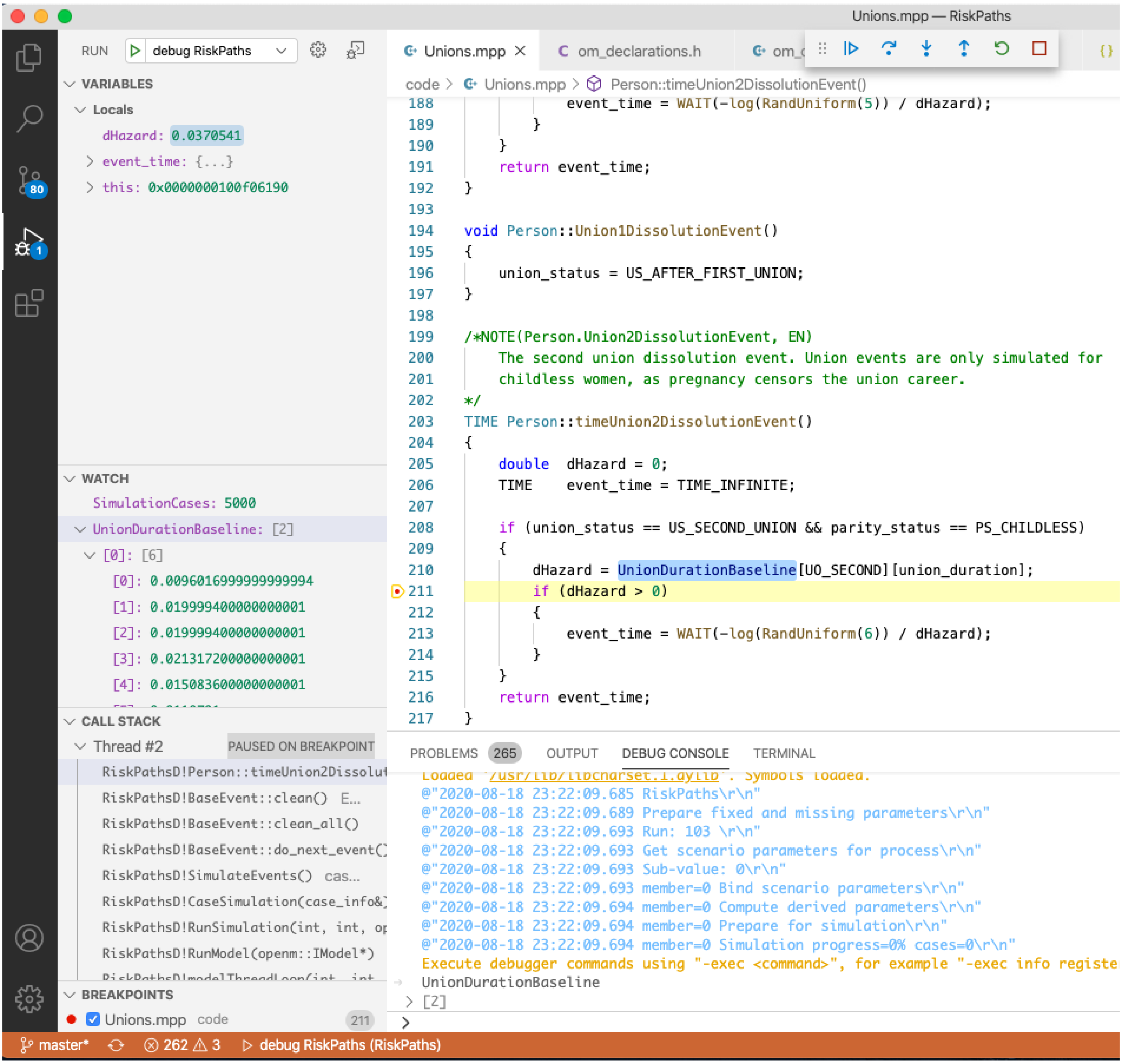
Set breakpoint(s):

- open any model.ompp or *.mpp file and put breakpoint in it

- (optional) RunSimulation entry point using File -> Open File... -> use/case_based/case_based_common.ompp -> RunSimulation()
- (optional) `main()` entry point: File -> Open File... -> openm/libopenm/main.cpp



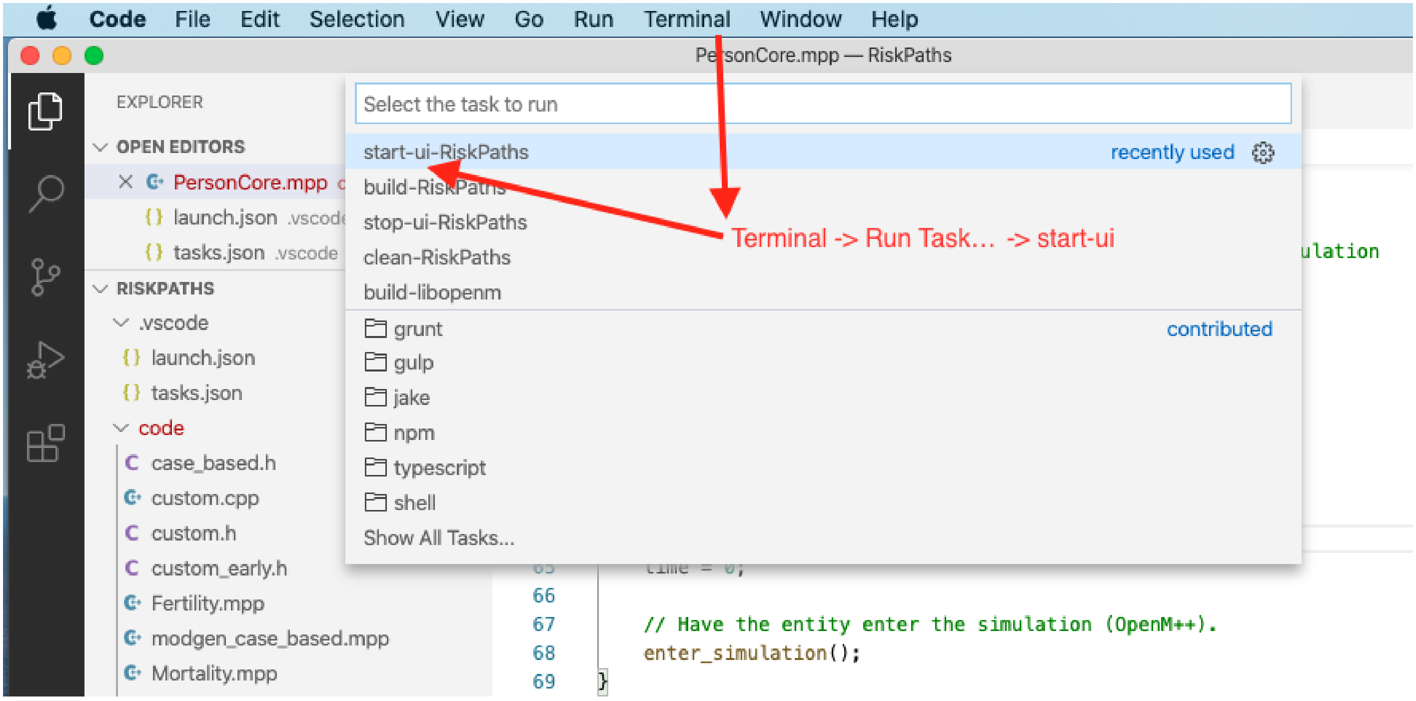
To inspect model parameters add Watch variable:

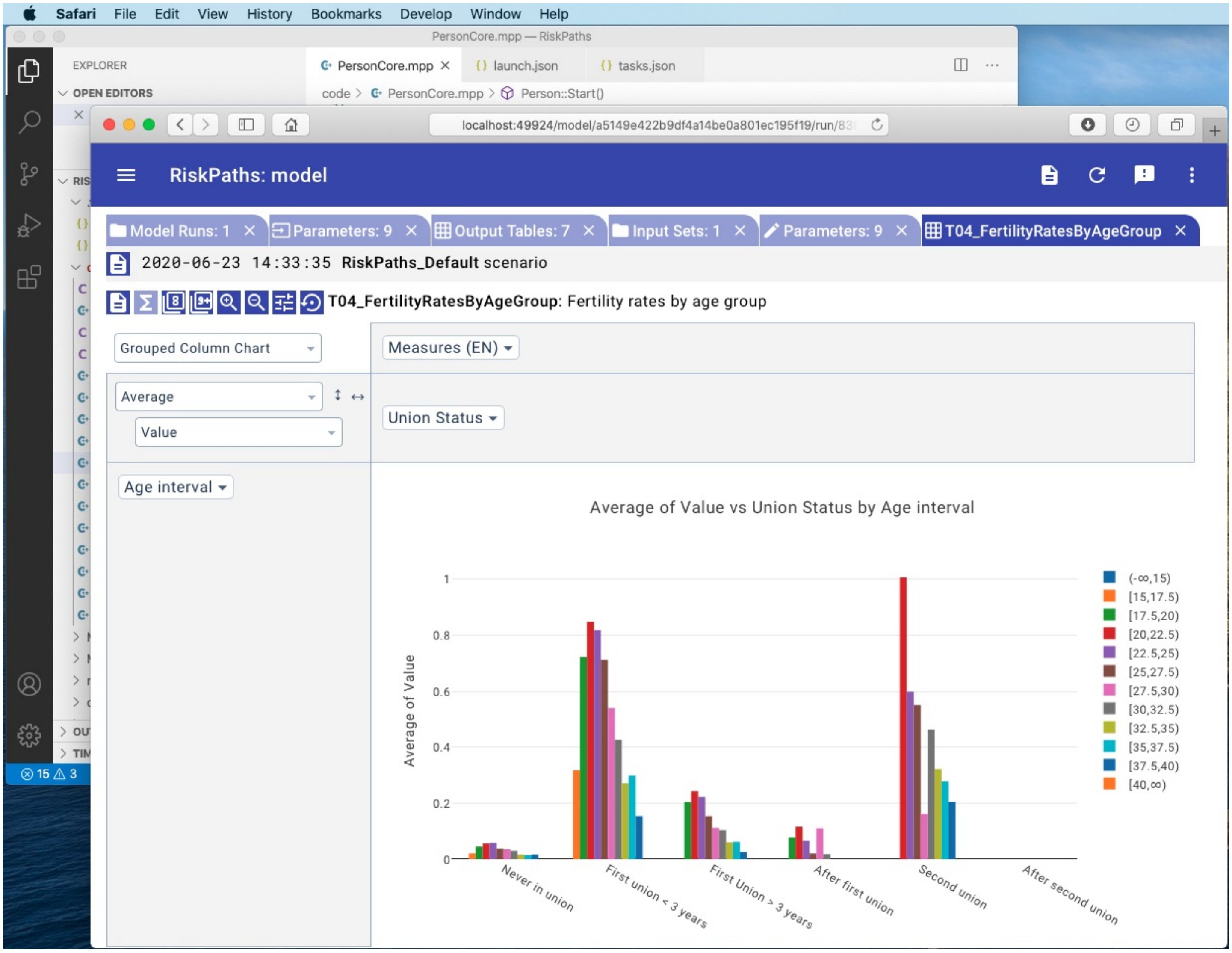


Start model UI on MacOS from VSCode

To start model UI from VSCode use menu: Terminal -> Run Tasks... -> start-ui-RiskPaths

To stop backgound `oms` web-service after you done with model UI use: Terminal -> Run Tasks... -> stop-ui-RiskPaths





Model run options

As described at Linux Quick Start for Model Users you can run the model with different options. For example, you can calculate 8 sub-values (a.k.a. sub-samples, members, replicas), use 4 threads and simulate 8000 cases:

```
./RiskPathsD -OpenM.SubValues 8 -OpenM.Threads 4 -Parameter.SimulationCases 8000
```

You can supply run options as model command line arguments or by using model.ini file:

```
[OpenM]
SubValues = 8
Threads = 4

[Parameter]
SimulationCases=8000
```

```
./RiskPathsD -ini RiskPathsD.ini
```

There are two possible ways to use model ini-file with Visual Studio Code:

- by adding `-ini RiskPaths.ini` command line argument to model executable. Go to menu -> Run -> Open Configurations and edit `launch.json` at `"program"` line:

  ```
  {
    // .... .... ....
    "program": "${workspaceFolder}/ompp-linux/bin/RiskPathsD -ini RiskPaths.ini",
    // .... .... ....
  }
  ```

- by adding `MODEL_INI=RiskPaths.ini` command line argument to model make. Go to menu -> Terminal -> Configure Task -> build-RiskPaths and edit `tasks.json` at `"command": "make ....` line:

  ```
  {
    "tasks": [
      {
        "label": "build-RiskPaths",
        "command": "make MODEL_INI=RiskPaths.ini all publish run",
      // .... .... ....
      }]
  }
  ```

  That `MODEL_INI` argument will be passed to model executable when `make` run the model as:

  ```
  ompp-linux/bin/RiskPathsD -ini RiskPaths.ini
  ```

MacOS: Create and Debug Models using Xcode

What do you need

- Download: latest binary files and source code

- Documentation:
  - MacOS Quick Start for Developers

  - MacOS: Create and Debug Models

Prerequisites

- Tested on: MacOS 10.15 Catalina And Big Sur >= 11.1.

- Install Xcode and command line developer tools, if not installed already by Xcode: `xcode-select --install`.

- Check if clang, make and sqlite3 are installed on your computer:

```
g++ --version
....
Apple clang version 11.0.0 (clang-1100.0.33.12)

make --version
....
GNU Make 3.81

sqlite3 --version
....
3.28.0 2019-04-15 14:49:49
```

- Download and unpack latest openM++ release using Safari or curl:

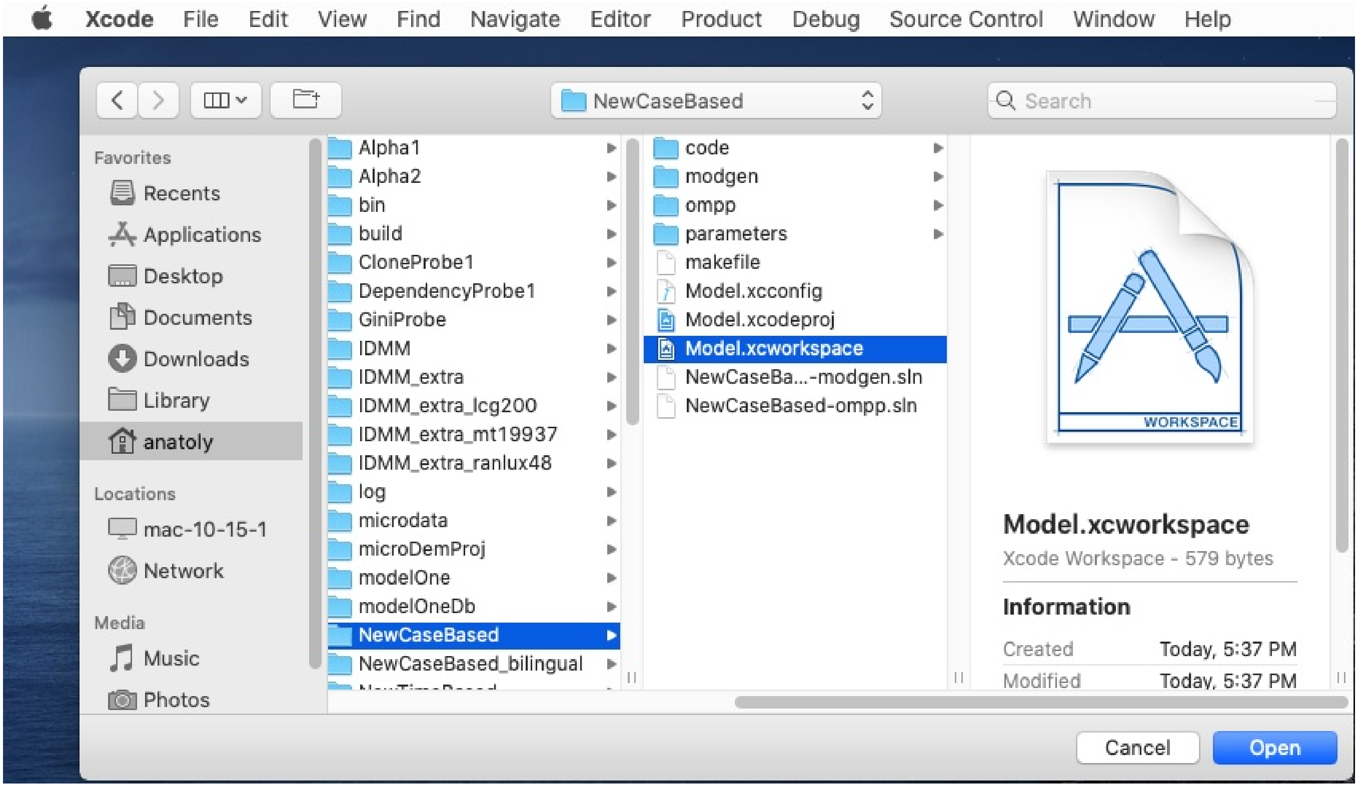
  ```
  curl -L -o om.tar.gz https://github.com/openmpp/main/releases/download/v1.6.0/openmpp_mac_20200621.tar.gz
  tar xzf om.tar.gz
  ```

Create Xcode project for new Model

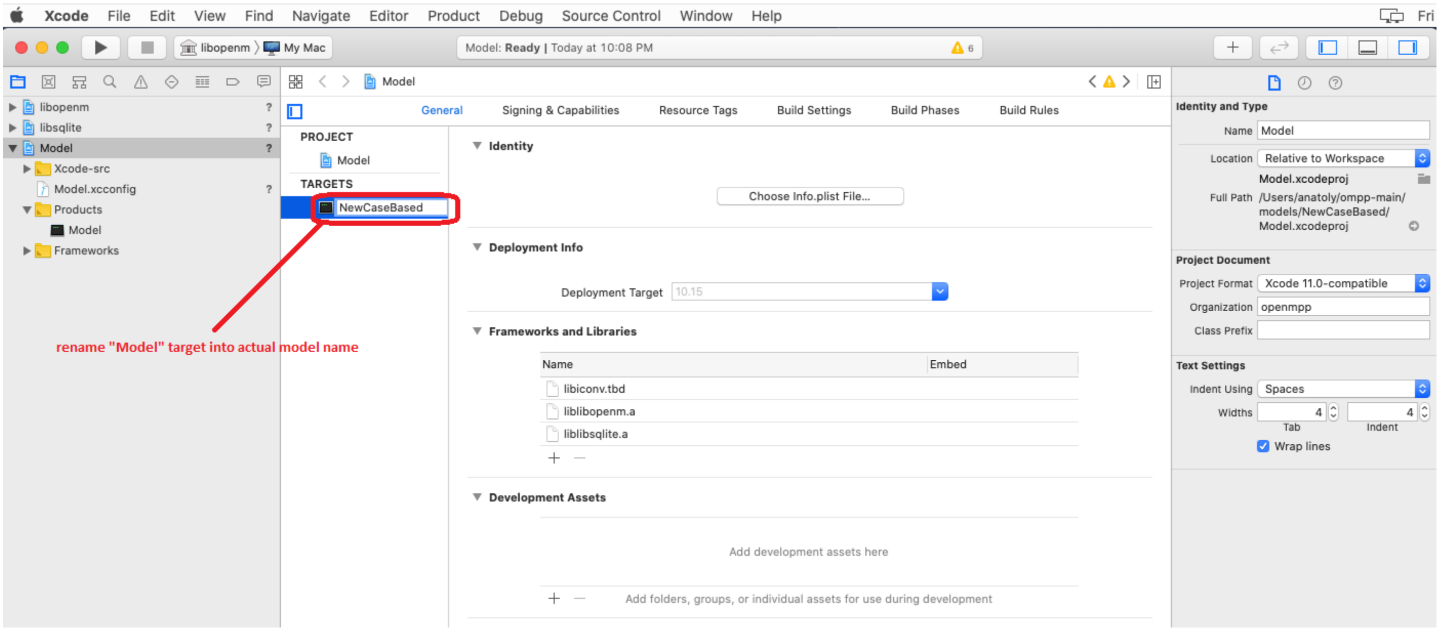
Copy model Xcode project files into your new "MyModel" directory, for example:

```
cd ~/openmpp_mac_20200621
cp -pr Xcode/Model.* models/MyModel/
```

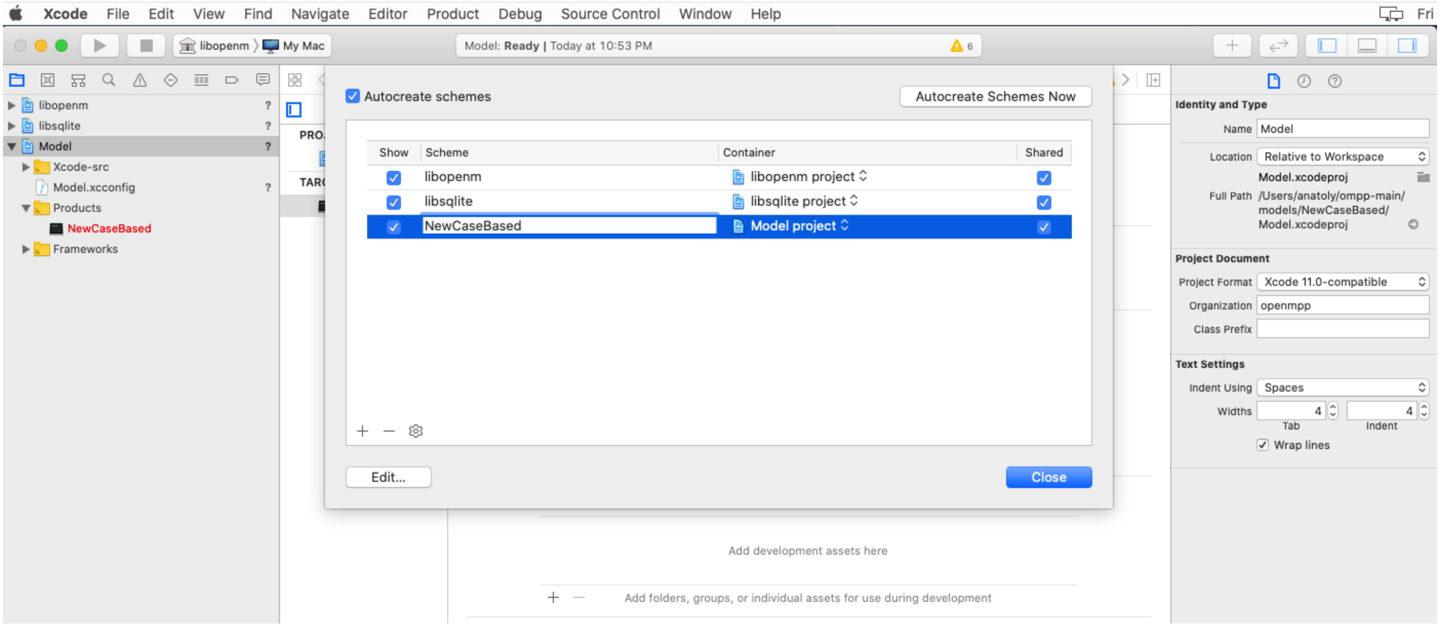
Start Xcode and open `~/openmpp_mac_20200621/models/MyModel/Model.xcworkspace`:



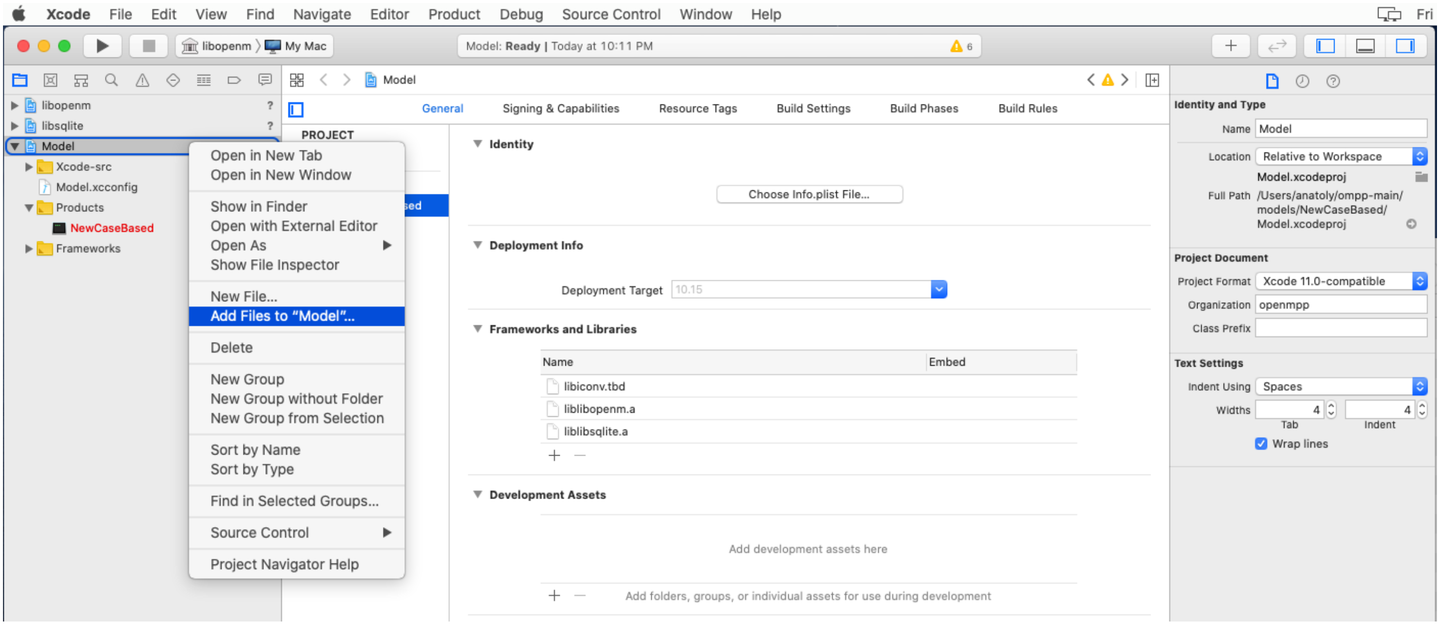
Rename model Project -> Targets -> click twice on target name -> and rename to MyModel

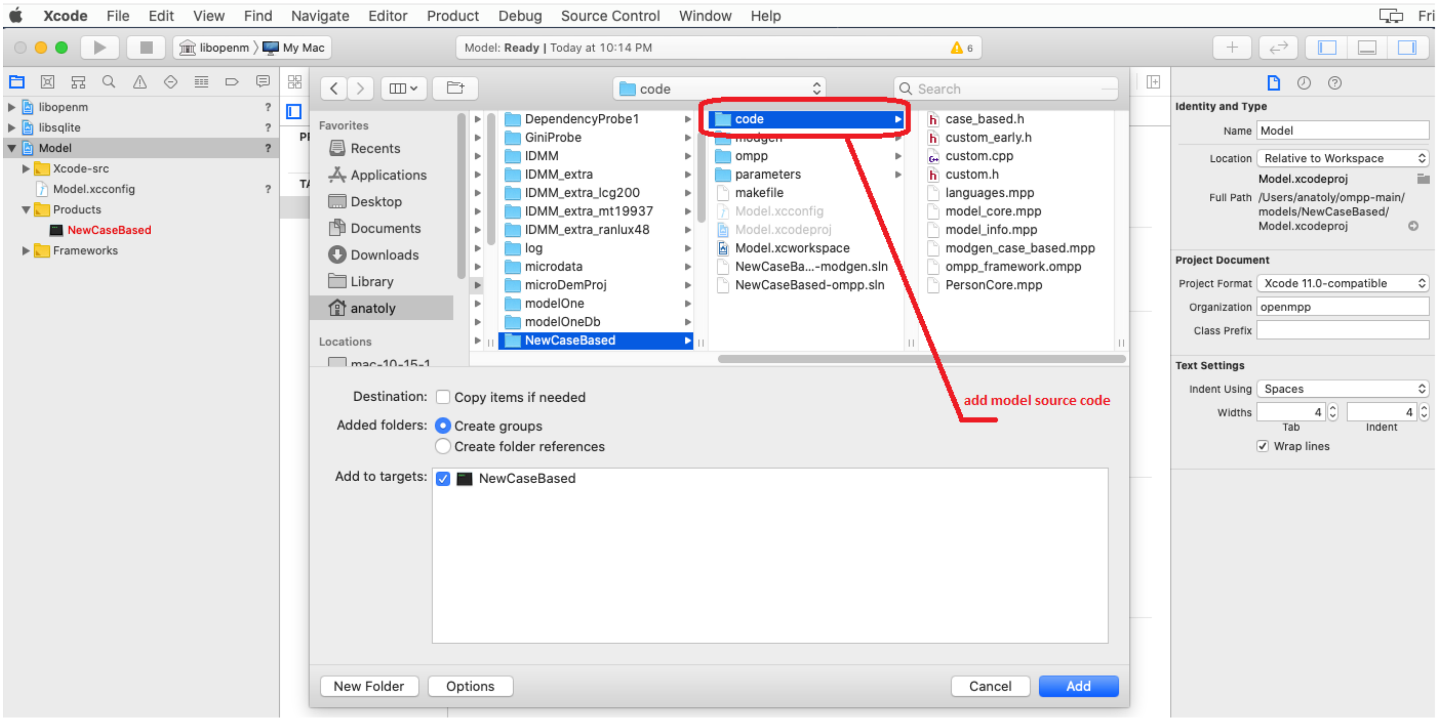


Rename model scheme using menu: Product -> Scheme -> Manage Schemes... -> click twice on "Model" scheme -> and rename to MyModel

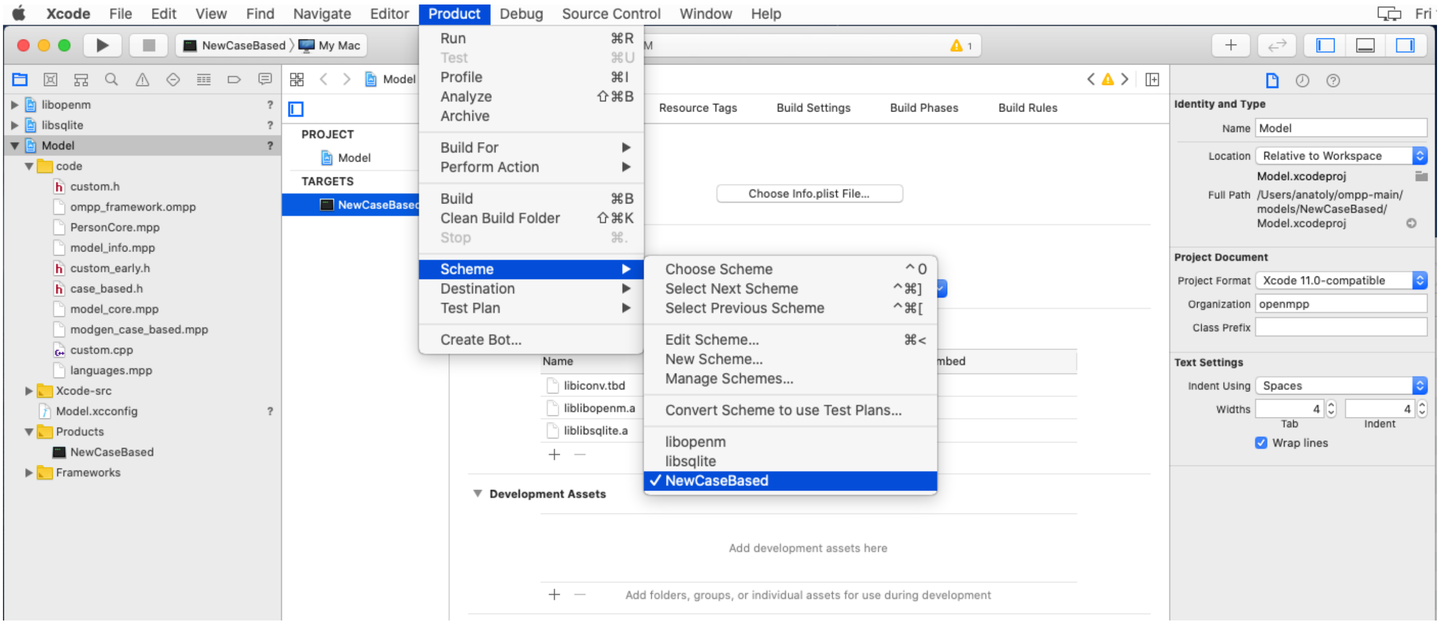


Add your model source code from `MyModel/code` folder:

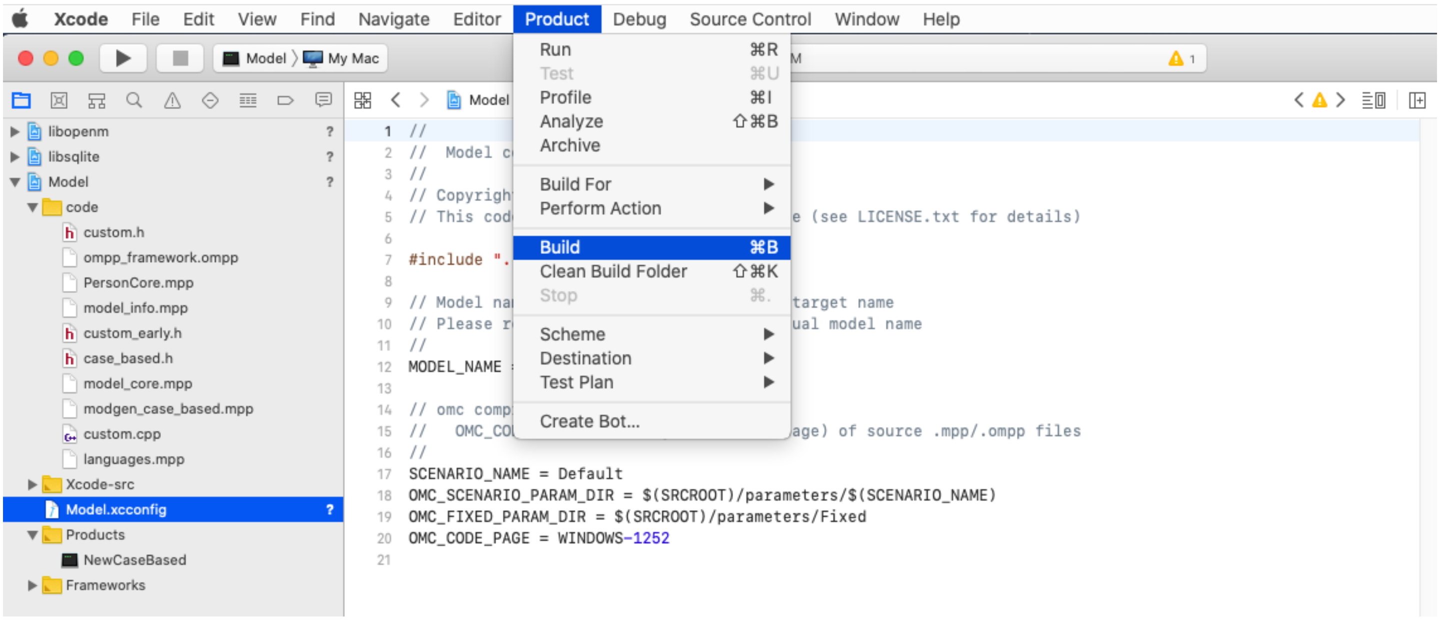




Make sure your model scheme selected by using menu: Product -> Scheme -> MyModel:

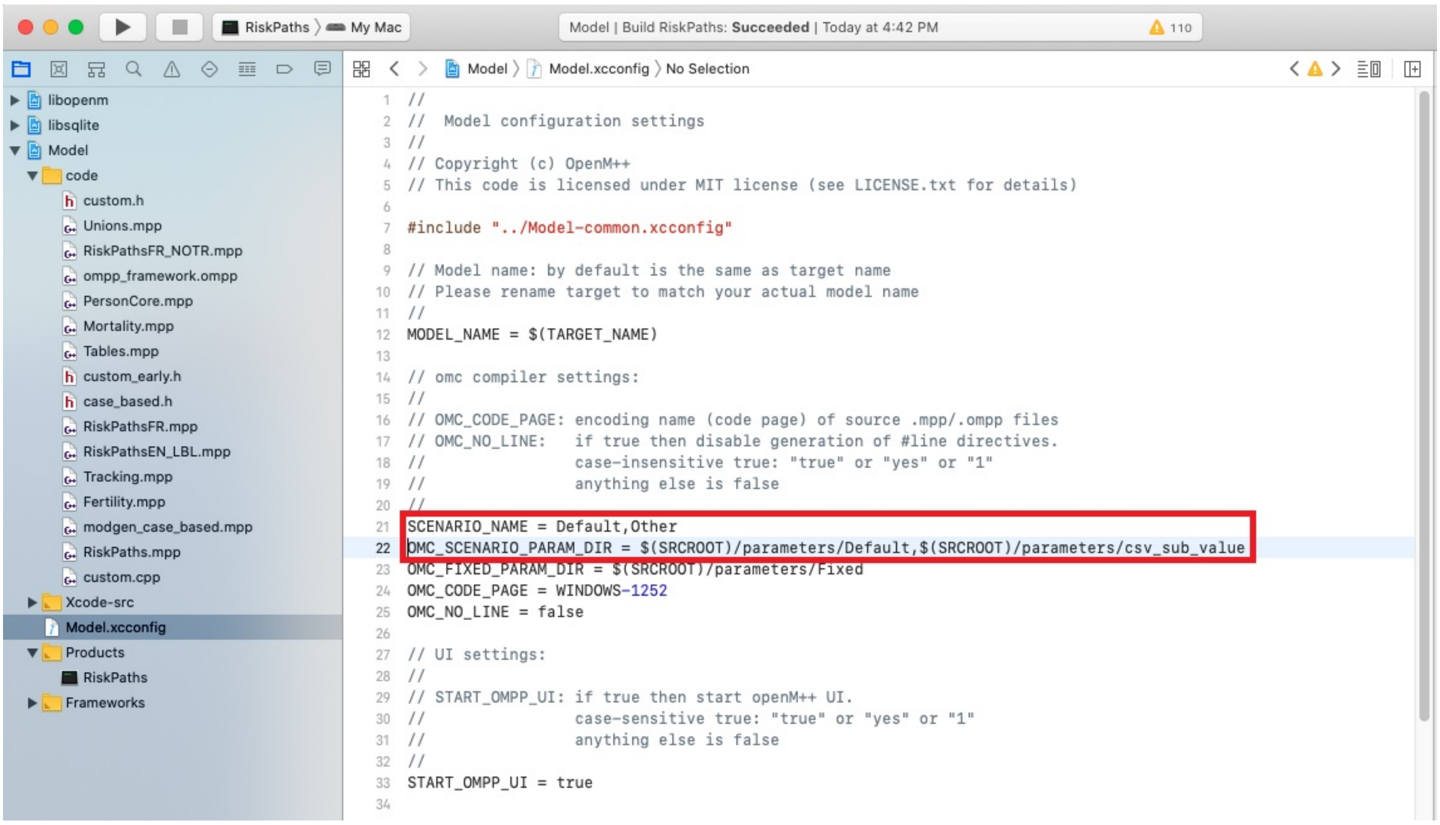


Build your model:



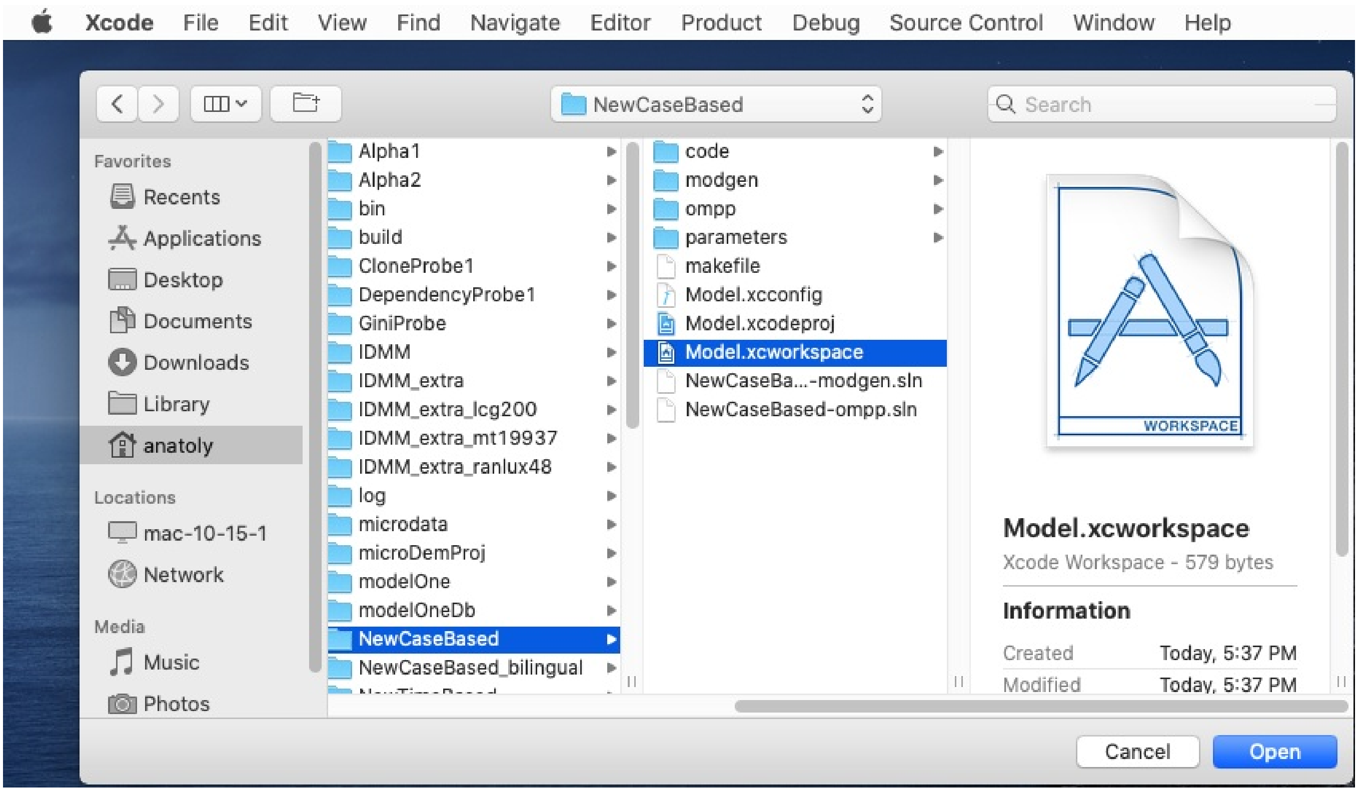
(Optional) Build the model with multiple scenarios:

- edit `Xcode-src/Model.xcconfig` and to specify additional scenario names and input directories, separated by comma. For example:
  - `SCENARIO_NAME = Default,Other`
  - `OMC_SCENARIO_PARAM_DIR = $(SRCROOT)/parameters/Default,$(SRCROOT)/parameters/SomeOther`



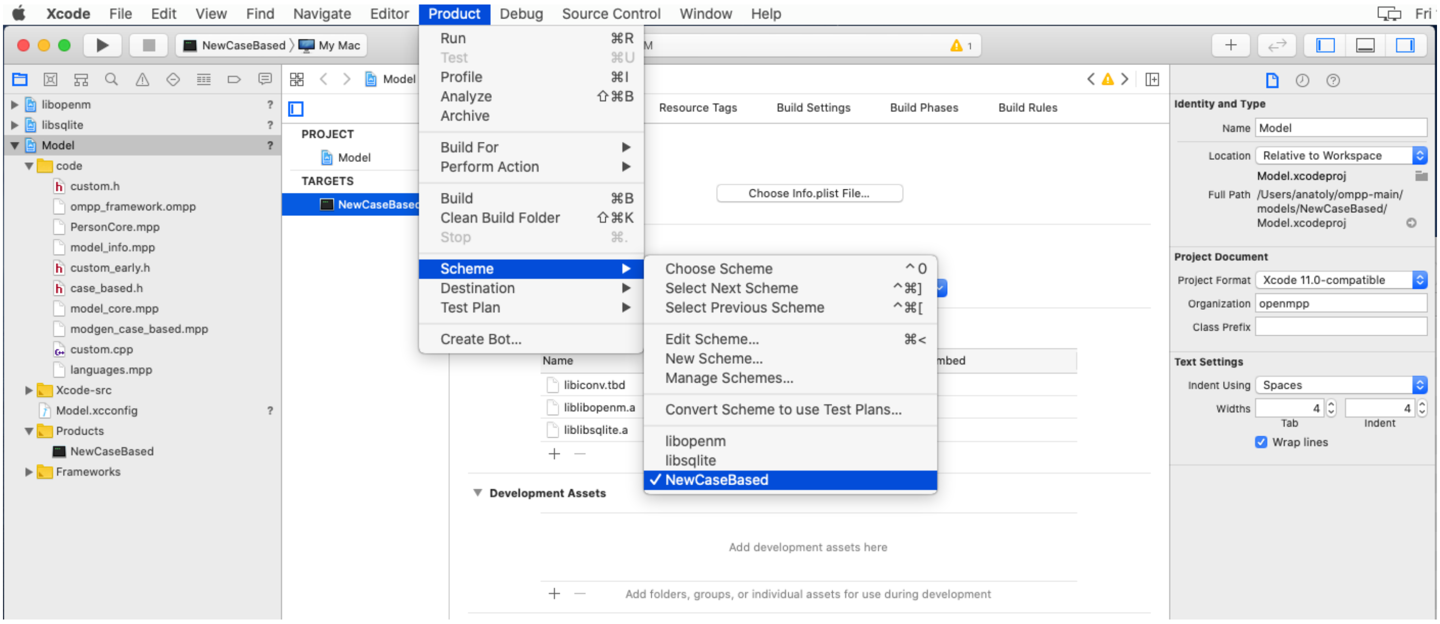
Debug openM++ Model using Xcode

Start Xcode and open your model workspace, for example: `~/openmpp_mac_20200621/models/MyModel/Model.xcworkspace`

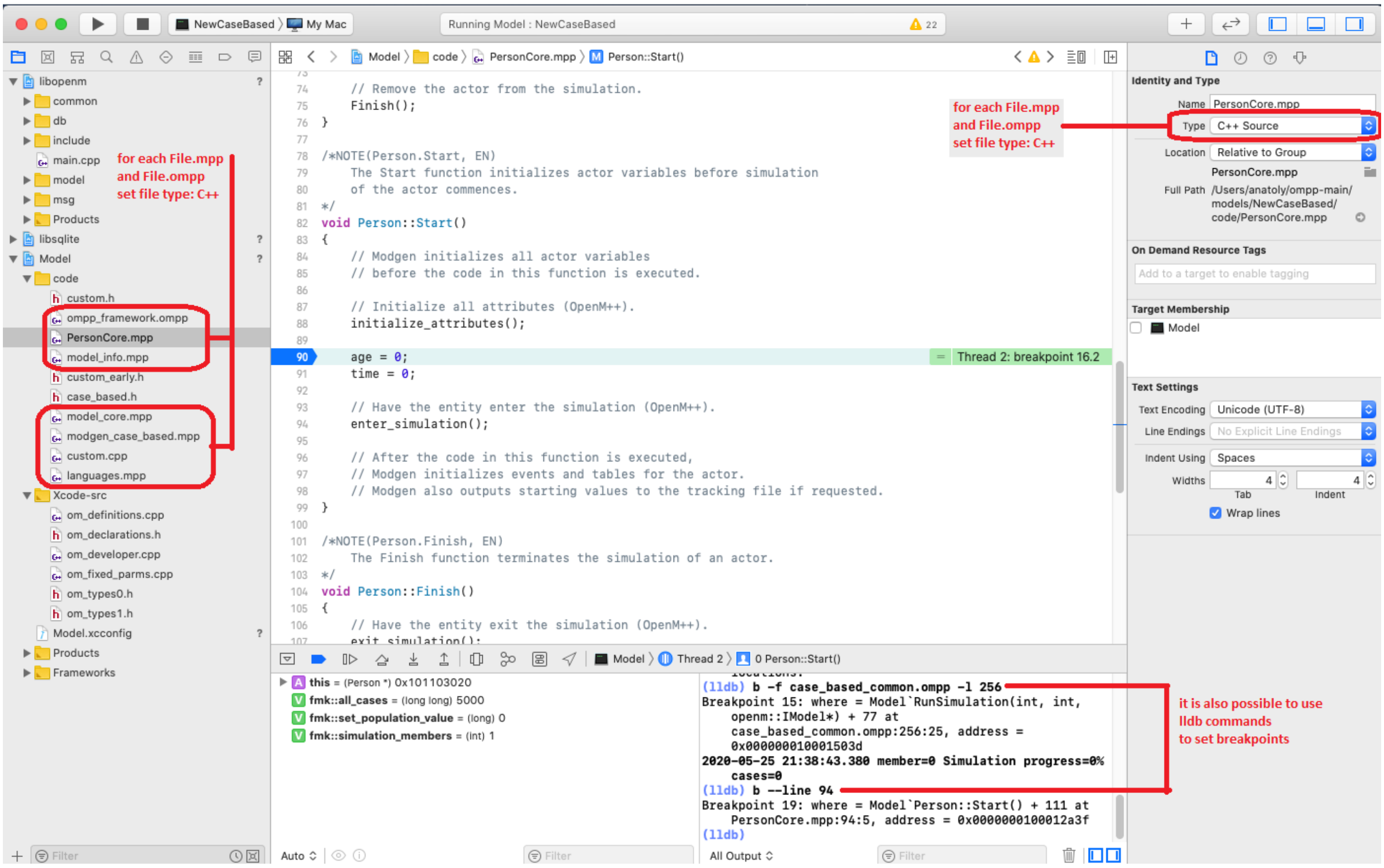


Use menu to select your model scheme: Product -> Scheme -> MyModel:

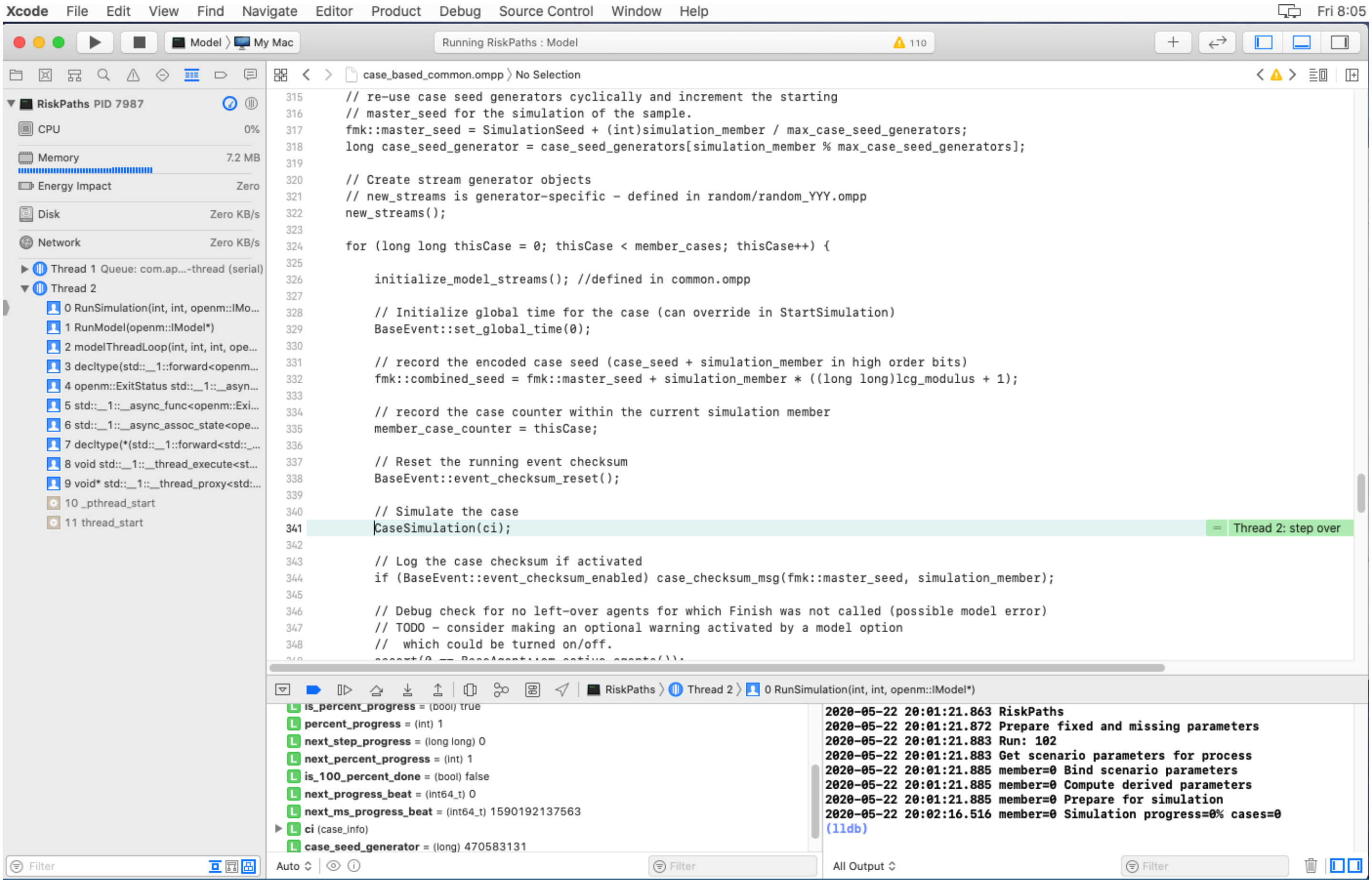
Known issue: Xcode UI may not update check mark on selected scheme. To fix it go to Product -> Scheme -> Manage Schemes... and use mouse to drag any scheme to move it up or down.



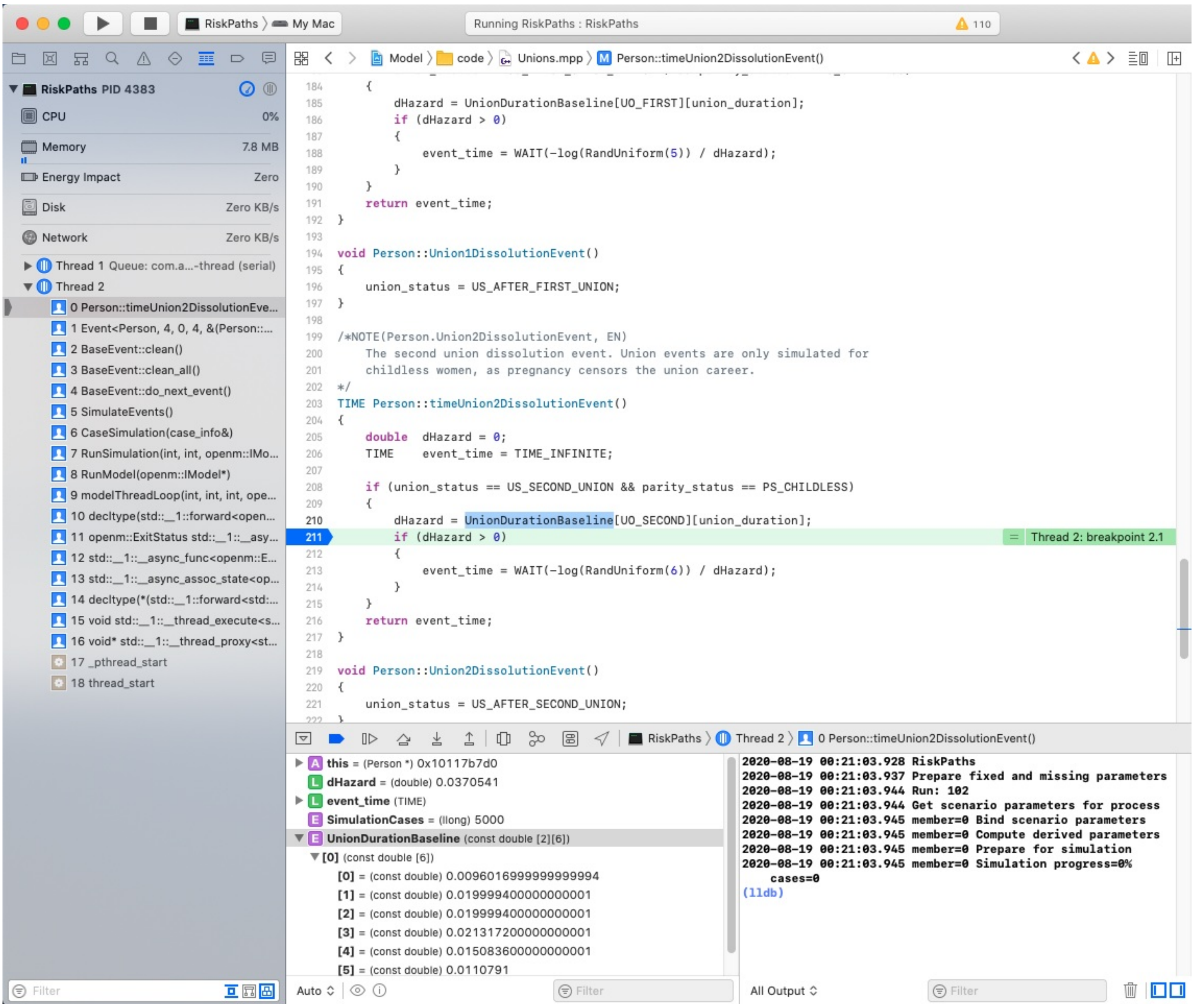
(Optional) If you want to set breakpoints in any *.mpp or* .ompp files then tell to Xcode it is "C++ Source" file(s):



Run and debug your openM++ model:

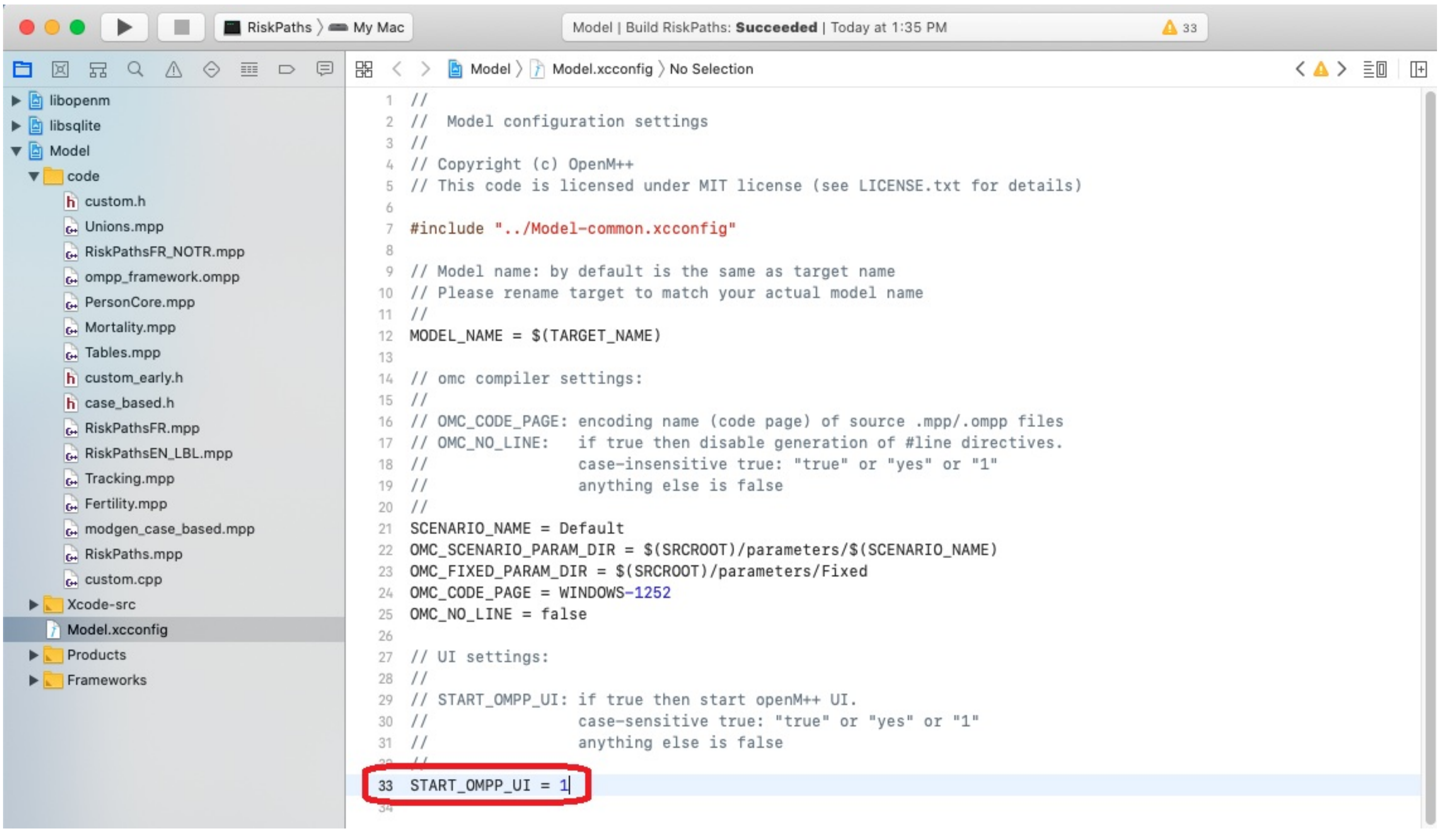


To inspect model parameters go to Debug Area and Add Expression:

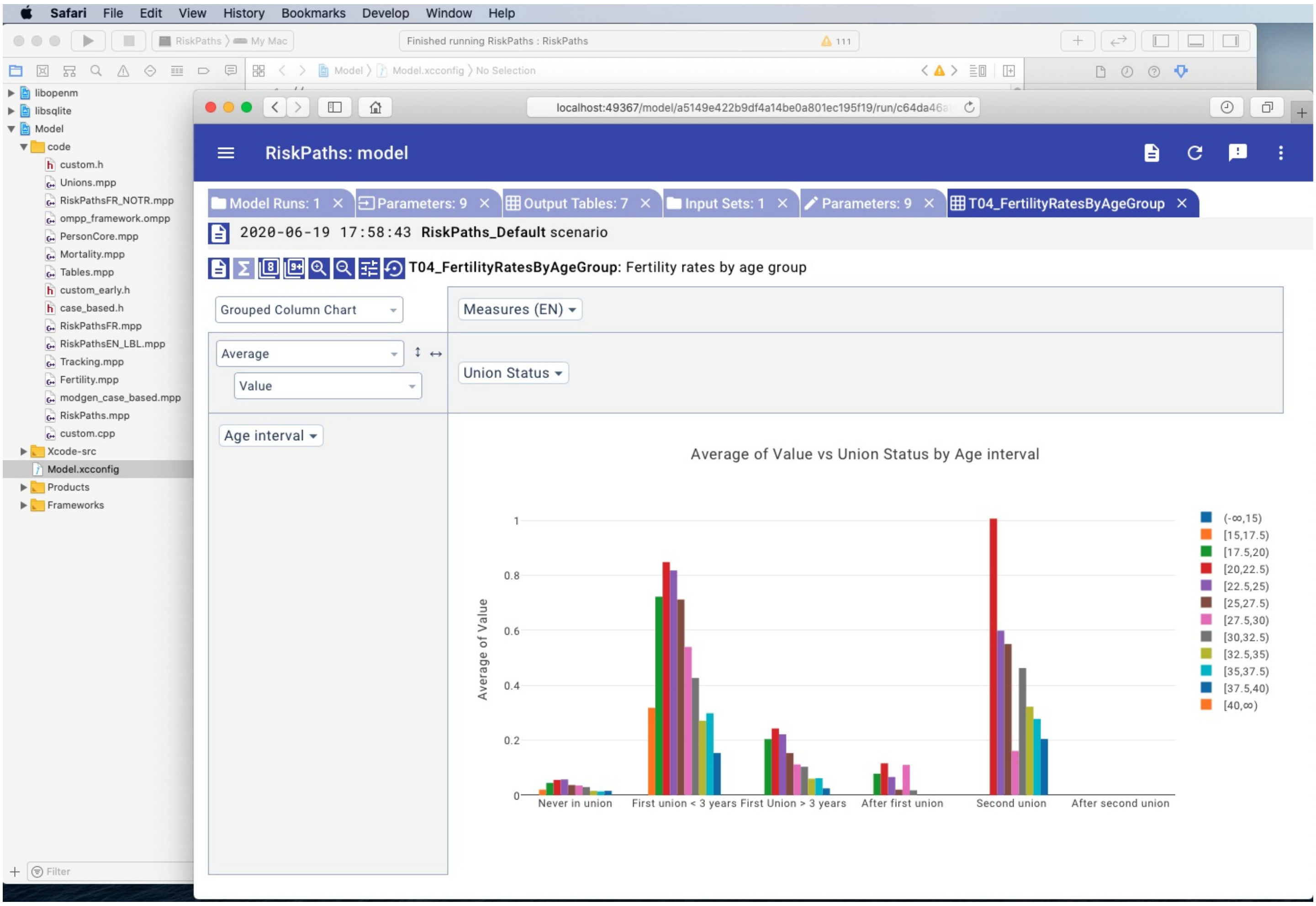


Start model UI on MacOS from Xcode

To start model UI after build completed please change `Model.xcconfig` variable `START_OMPP_UI` to "1" or "true" or "yes" (case-sensitive)



RiskPaths
My Mac
Model | Build RiskPaths: Succeeded | Today at 1:35 PM
33
Model
Model.xcconfig
No Selection
libopenm
libsqlite
Model
code
custom.h
Unions.mpp
RiskPathsFR_NOTR.mpp
ompp_framework.ompp
PersonCore.mpp
Mortality.mpp
Tables.mpp
custom_early.h
case_based.h
RiskPathsFR.mpp
RiskPathsEN_LBL.mpp
Tracking.mpp
Fertility.mpp
modgen_case_based.mpp
RiskPaths.mpp
custom.cpp
Xcode-src
Model.xcconfig
Products
Frameworks
//
// Model configuration settings
//
// Copyright (c) OpenM++
// This code is licensed under MIT license (see LICENSE.txt for details)
#include "../Model-common.xcconfig"
// Model name: by default is the same as target name
// Please rename target to match your actual model name
//
MODEL_NAME = $(TARGET_NAME)
// omc compiler settings:
//
// OMC_CODE_PAGE: encoding name (code page) of source .mpp/.ompp files
// OMC_NO_LINE: if true then disable generation of #line directives.
//   case-insensitive true: "true" or "yes" or "1"
//   anything else is false
//
SCENARIO_NAME = Default
OMC_SCENARIO_PARAM_DIR = $(SRCROOT)/parameters/$(SCENARIO_NAME)
OMC_FIXED_PARAM_DIR = $(SRCROOT)/parameters/Fixed
OMC_CODE_PAGE = WINDOWS-1252
OMC_NO_LINE = false
// UI settings:
//
// START_OMPP_UI: if true then start openM++ UI.
//   case-sensitive true: "true" or "yes" or "1"
//   anything else is false
START_OMPP_UI = 1



Safari
File
Edit
View
History
Bookmarks
Develop
Window
Help
Finished running RiskPaths : RiskPaths
localhost:49367/model/a5149e422b9df4a14be0a801ec195f19/run/c64da46
RiskPaths: model
Model Runs: 1
Parameters: 9
Output Tables: 7
Input Sets: 1
Parameters: 9
T04_FertilityRatesByAgeGroup
2020-06-19 17:58:43 RiskPaths_Default scenario
T04_FertilityRatesByAgeGroup: Fertility rates by age group
Grouped Column Chart
Measures (EN)
Average
Value
Union Status
Age interval
Average of Value vs Union Status by Age interval
Average of Value
Never in union
First union < 3 years
First Union > 3 years
After first union
Second union
After second union
(-∞,15)
[15,17.5)
[17.5,20)
[20,22.5)
[22.5,25)
[25,27.5)
[27.5,30)
[30,32.5)
[32.5,35)
[35,37.5)
[37.5,40)
[40,∞)
Filter

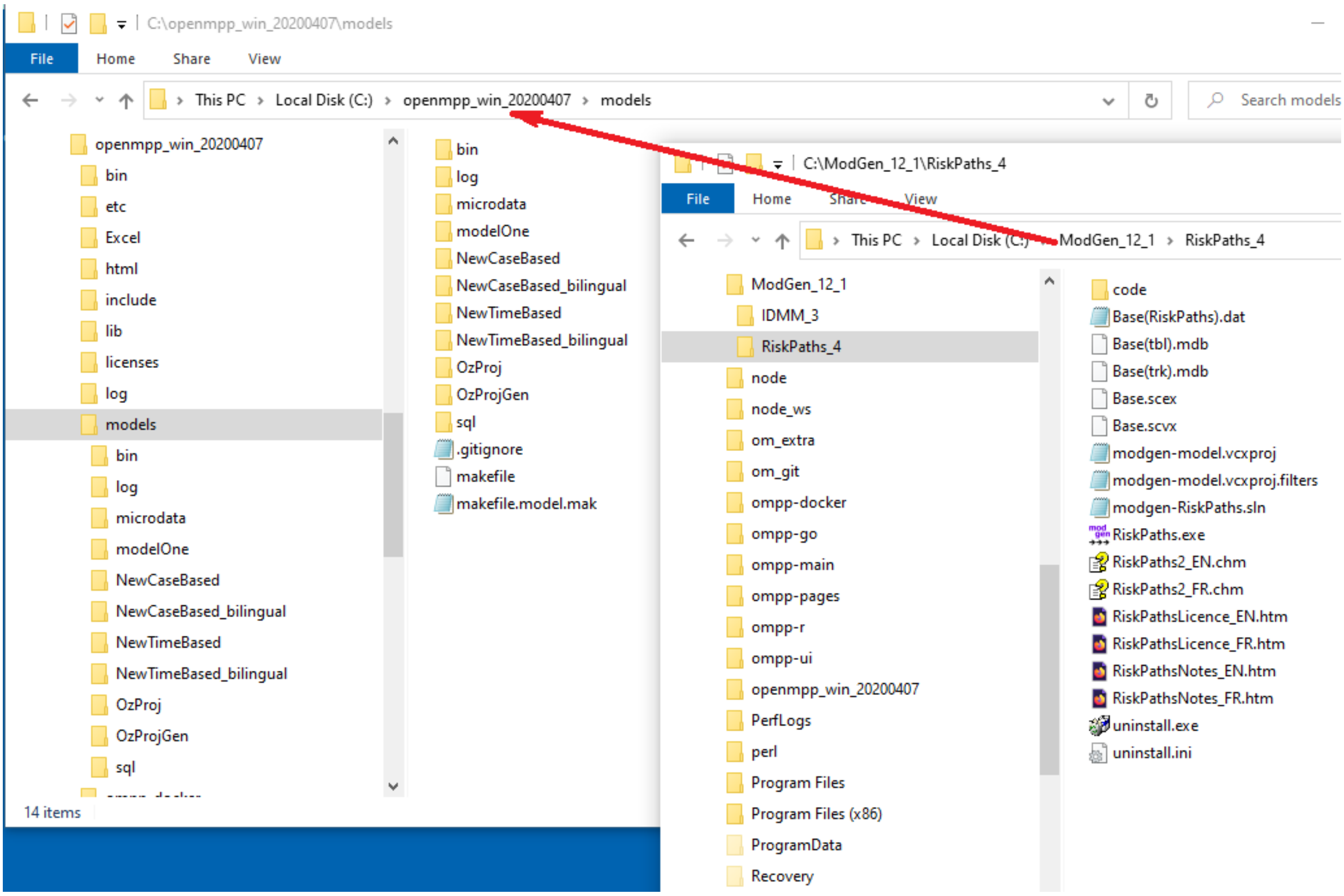
Modgen: Convert case-based model to openM++

Overview

OpenM++ provides superset of Modgen language specification and therefore able to compile Modgen source files. Conversion from Modgen include following:

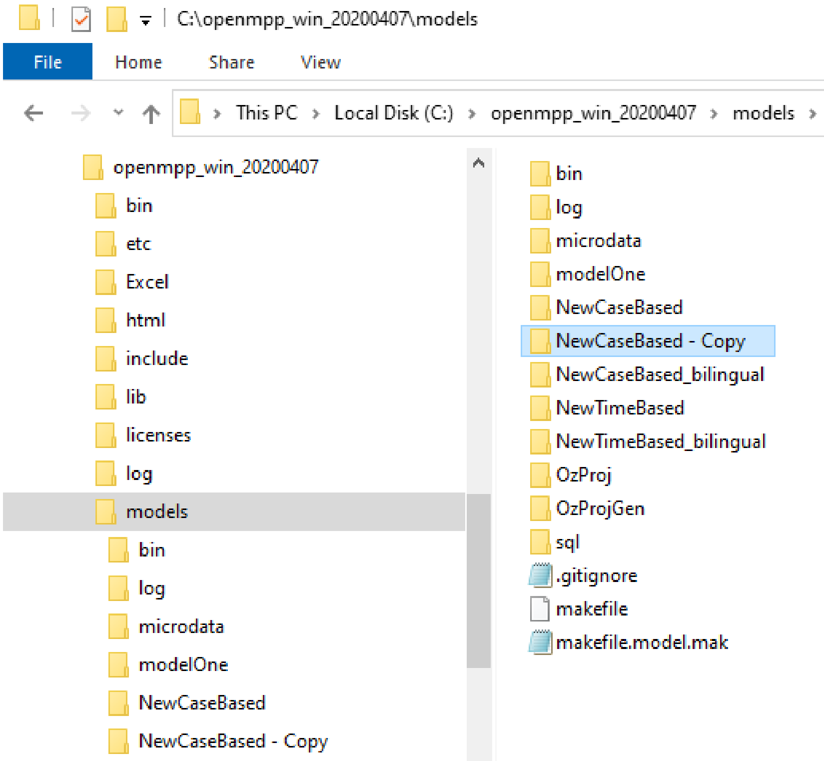
- Make sure you are done with: Windows: Quick Start for Model Developers
- Clone existing openM++ case-based model, for example: NewCaseBased
- Rename model directory and solution to YourModelName, for example: RiskPaths
- Replace NewCaseBased *.mpp modules with your model RiskPaths* .mpp files and inspect your code for any quirks (often none)
- Replace NewCaseBased *.dat parameter data with your model RiskPaths* .dat files
- Open Visual Studio, build the model and fix errors if necessary
- Run the model and verify simulation results

Below is step-by-step example how to convert RiskPaths model from Modgen 12.1 to openM++.



Clone existing openM++ model

As starting point please copy one of openM++ sample models, for case-based model we can start from NewCaseBased.

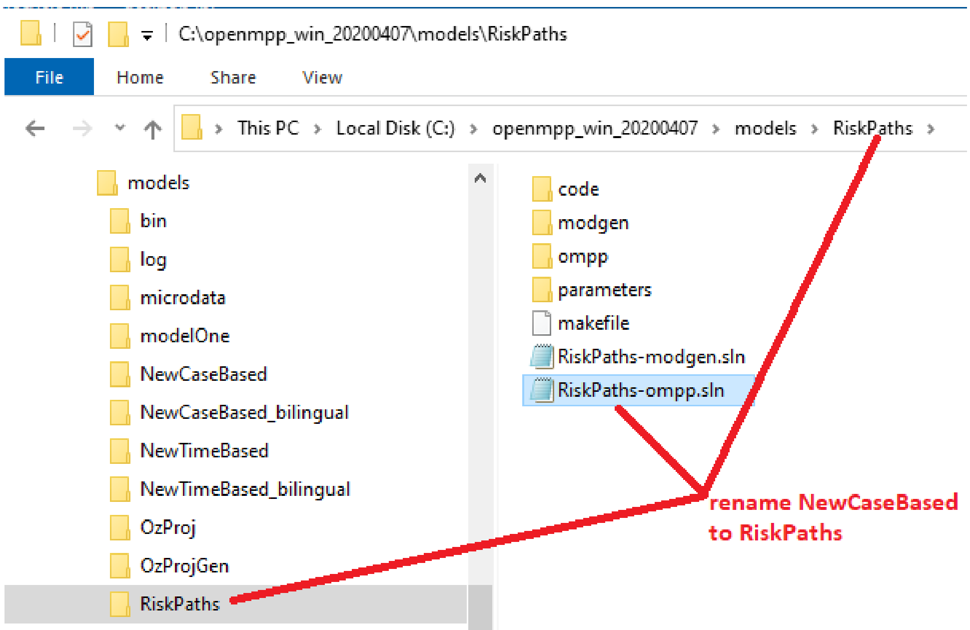


Rename model directory and solution

Rename directory and model solution into `YourModelName.sln`:

- rename `NewCaseBased - Copy` directory into `RiskPaths`
- rename `NewCaseBased-ompp.sln` into `RiskPaths-ompp.sln`
- (optional) rename `NewCaseBased-modgen.sln` into `RiskPaths-modgen.sln`

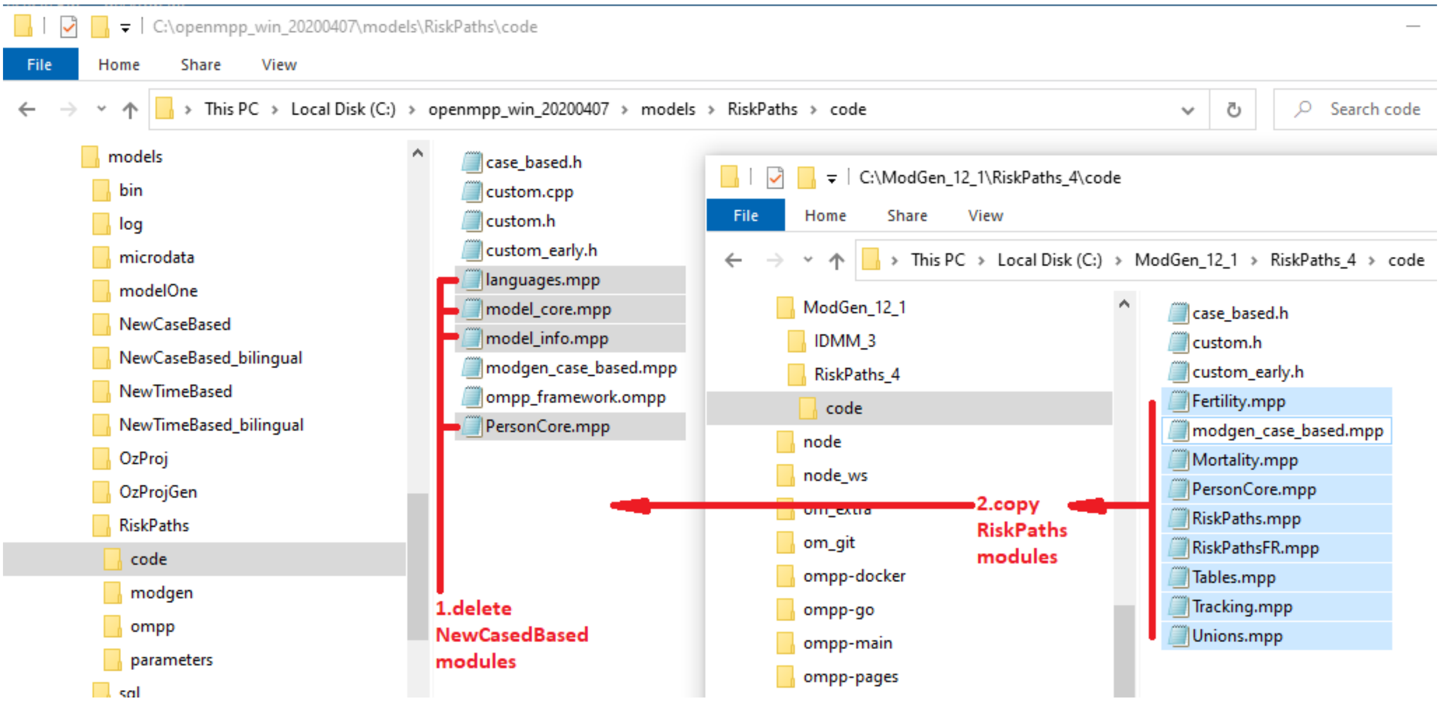
Note: It is not required to use model name as directory name and solution name, but it is openM++ convention and significantly simplifies model maintenance.

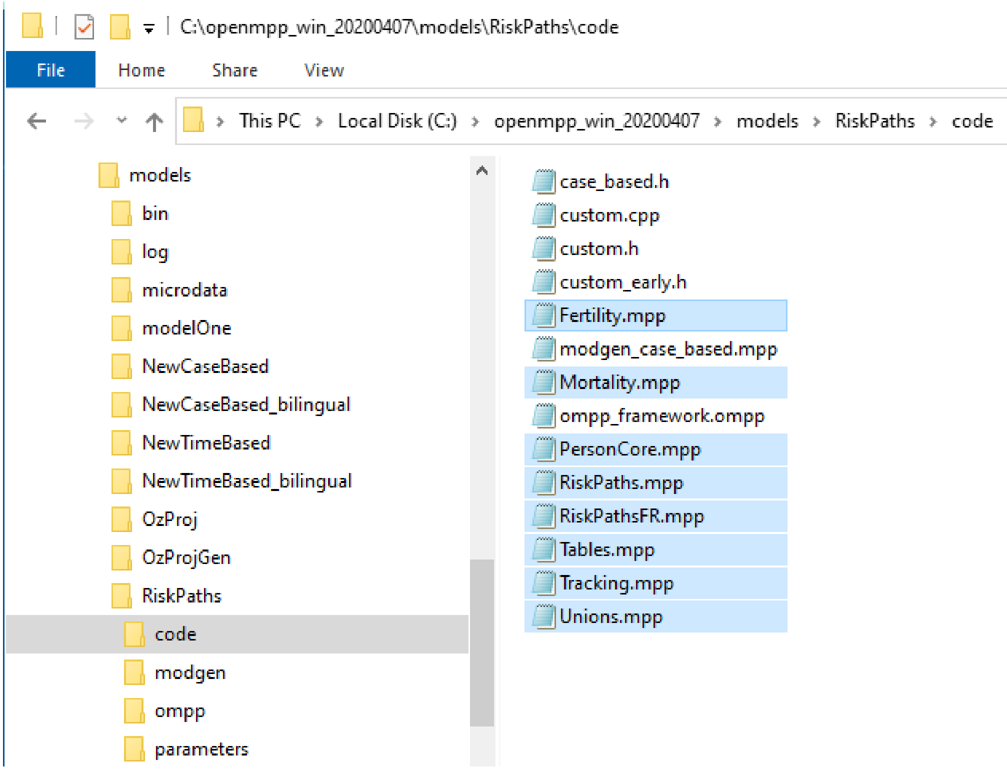


Replace sample model *.mpp modules with your model* .mpp files

Delete NewCaseBased *.mpp modules and copy your model substantive* .mpp files instead. For complex models with long maintenance history it may be not always easy to understand what *.mpp files are "substantive" and you may need to repeat this step multiple times.

It is also rare, but possible for some *.mpp modules to contain special quirky code in order to overcome issues in old version of Modgen or c++. Please inspect your code and adjust it, if necessary, to comply with c++17 standard.



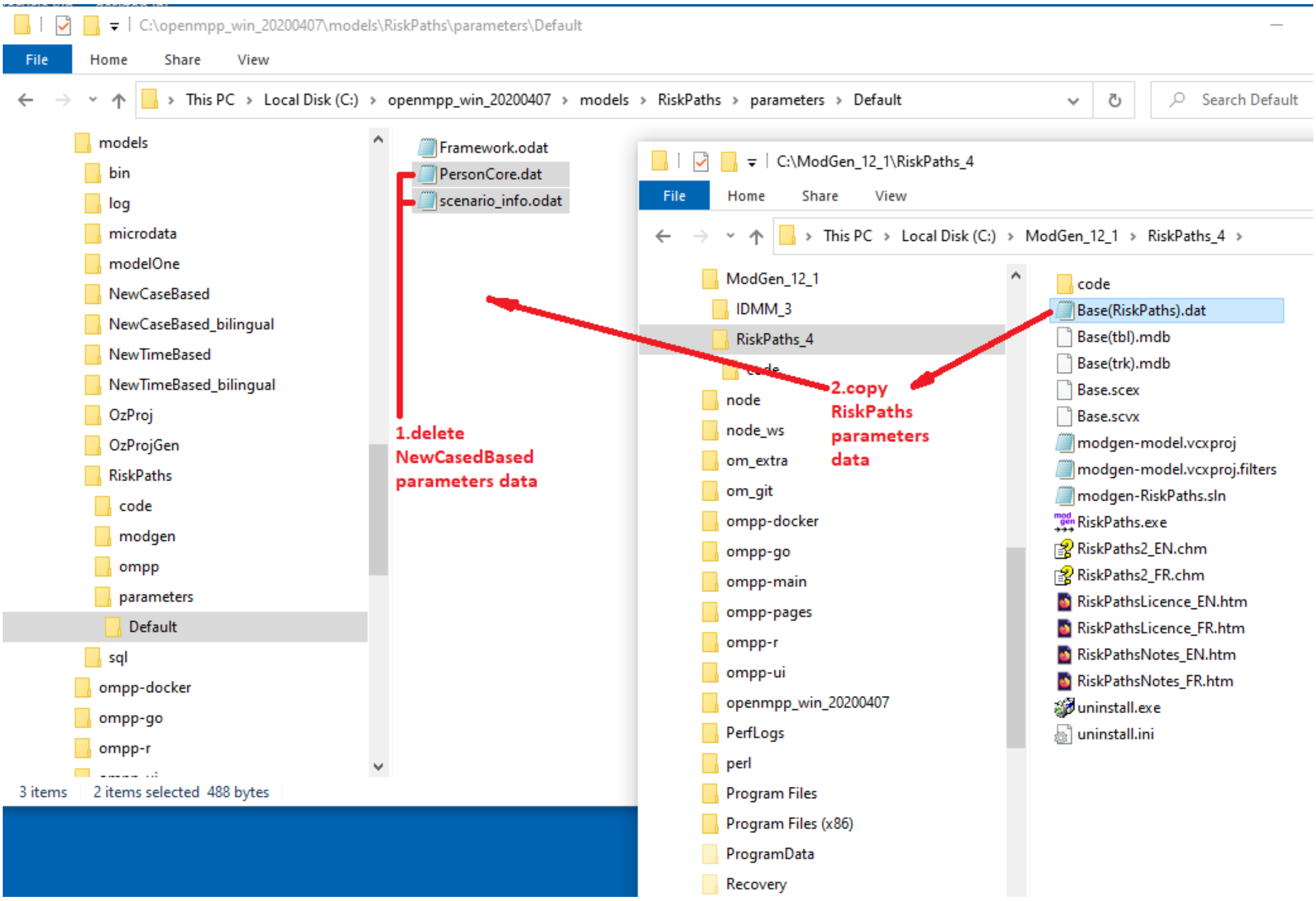


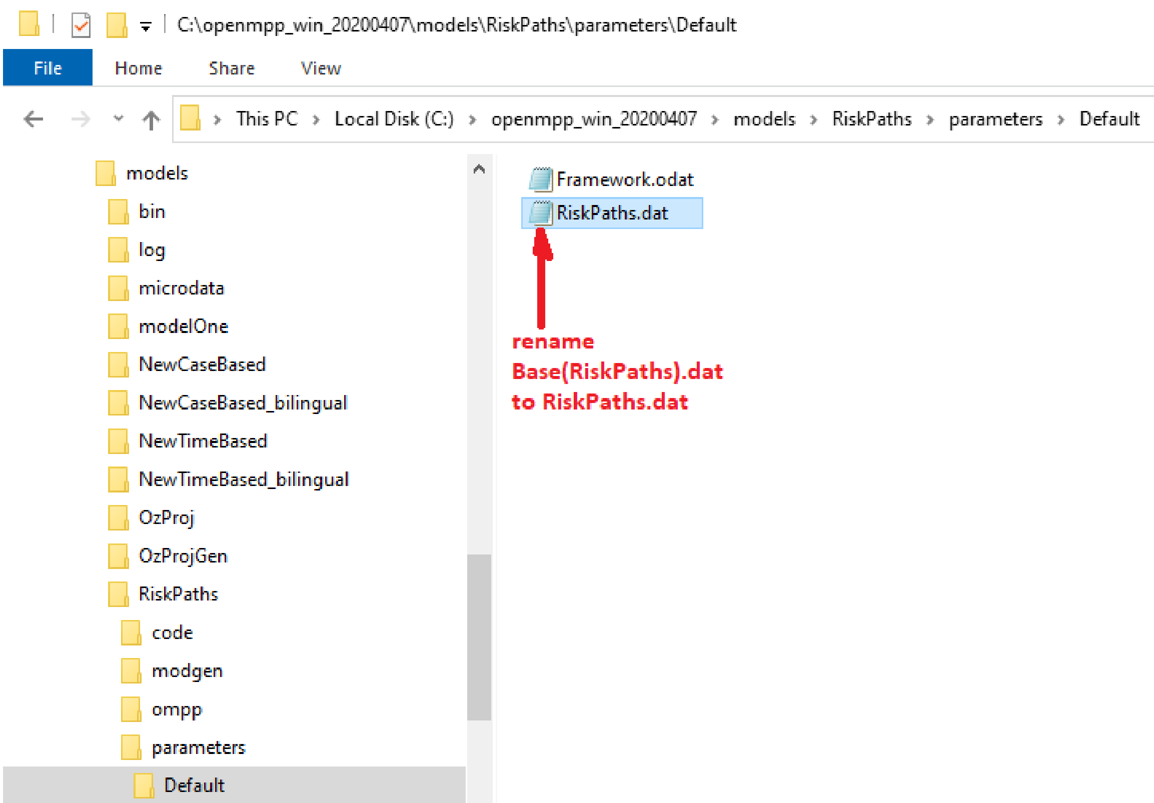
Replace sample model parameter data with your model *.dat files

For our example we need to:

- delete NewCaseBased `parameters/Default/PersonCore.dat` and `parameters/Default/scenario_info.odat`
- copy `Base(RiskPaths).dat`
- (optional) rename it into `RiskPaths.dat`

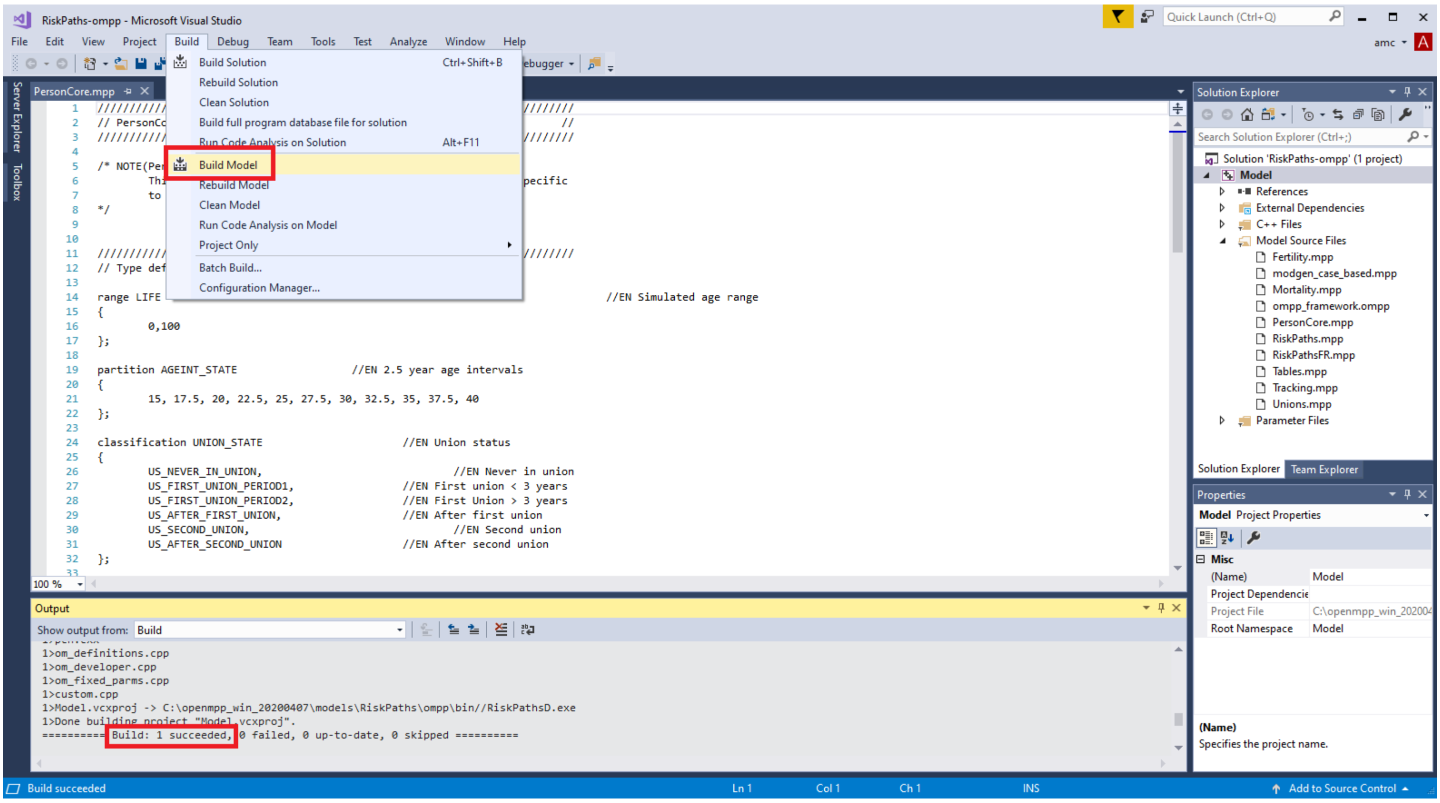
For complex models it is also possible to have `Fixed` parameters data. Please copy it into `parameters/Fixed/` sub-folder.





Open Visual Studio solution and build the model

Open `RiskPaths-ompp.sln` solution in Visual Studio and build the model, fix errors, if necessary.



Run the model and verify simulation results

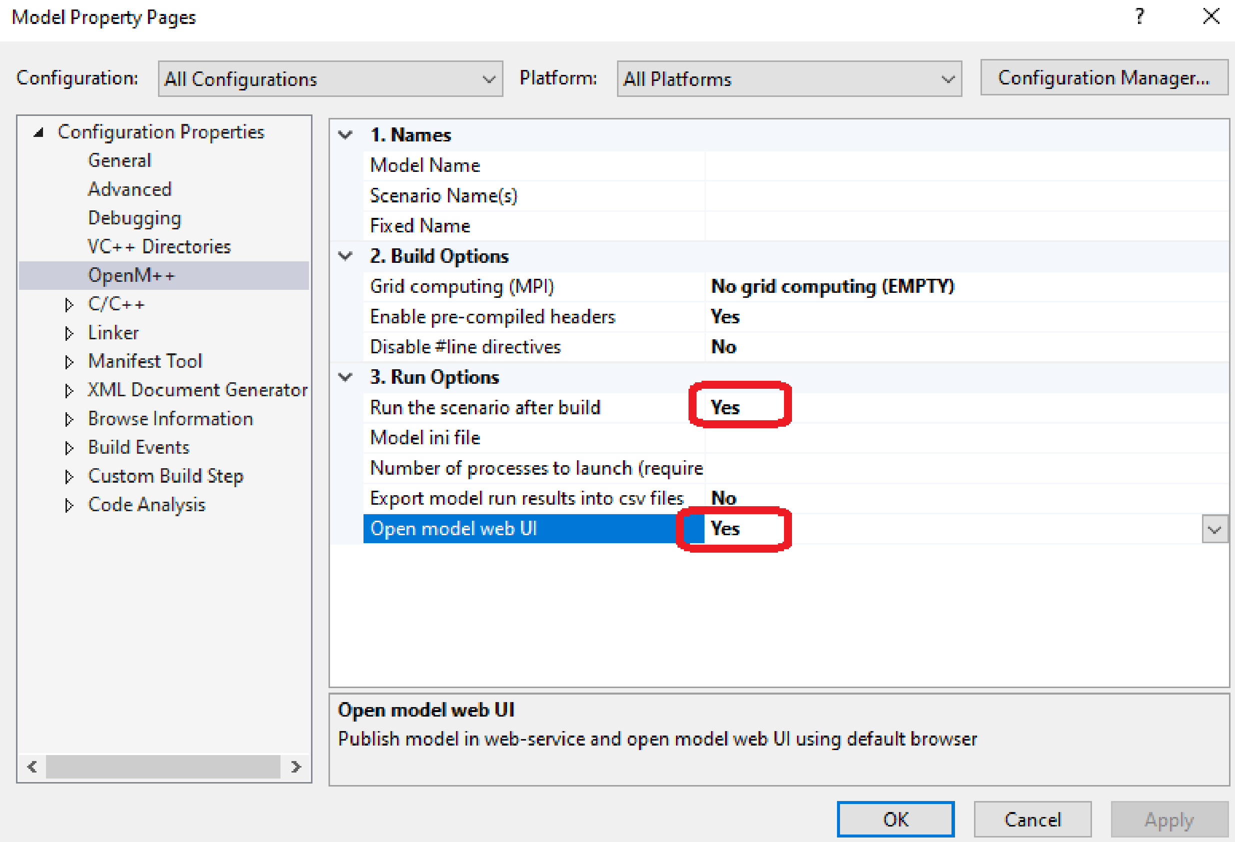
Last, but obviously very important step, is to run the model and compare Modgen and openM++ simulation results.

Check `parameters/Default/Framework.odat` values:

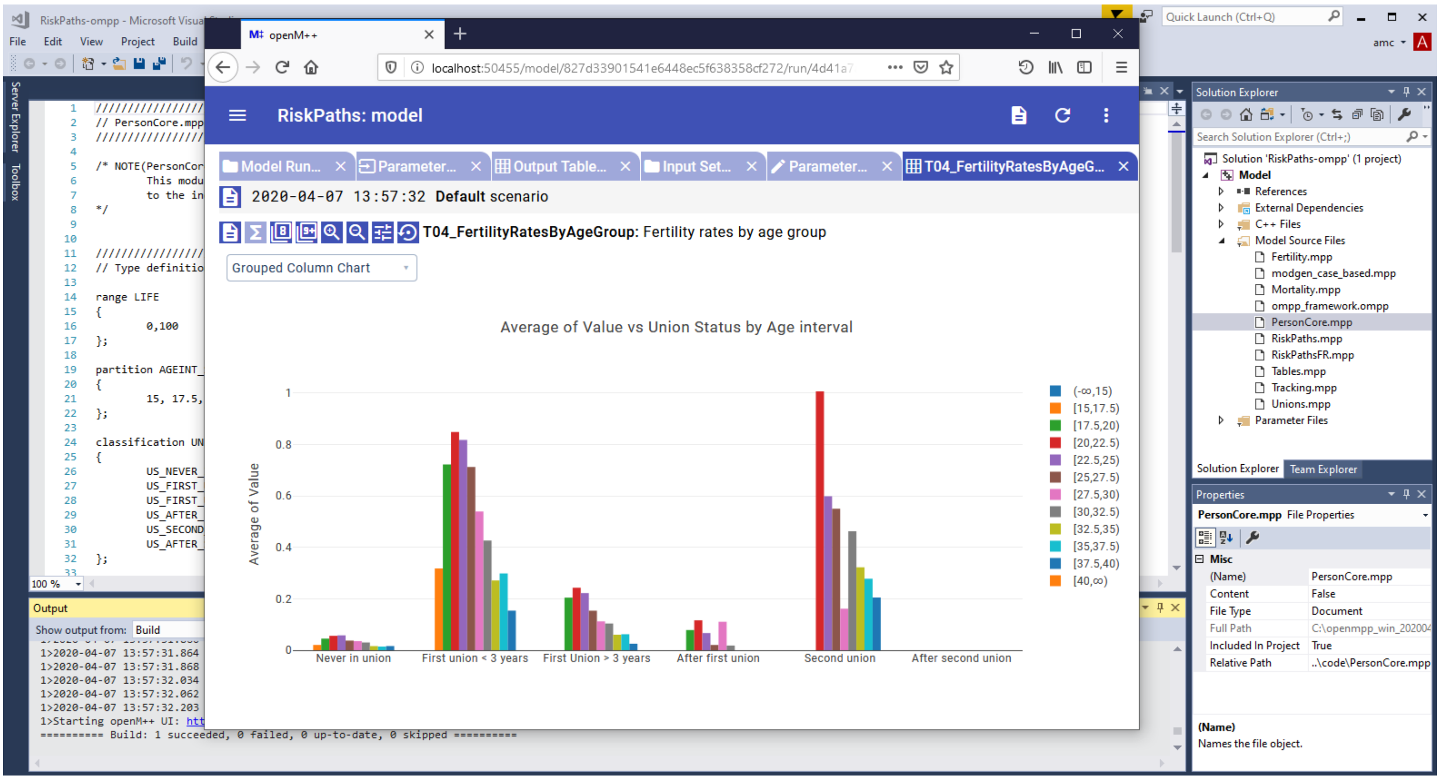
```
parameters {
  int SimulationSeed = 16807;
  long long SimulationCases = 5000;
};
```

and adjust number of simulation cases if required, re-build the model to update `SimulationCases` value in `RiskPaths.sqlite` model database.

You can run openM++ model from command line, or from Visual Studio by changing `Project -> Properties -> OpenM++ -> Run Options` :



It is possible to open model run results in openM++ UI (beta version) to examine model parameters and output results:



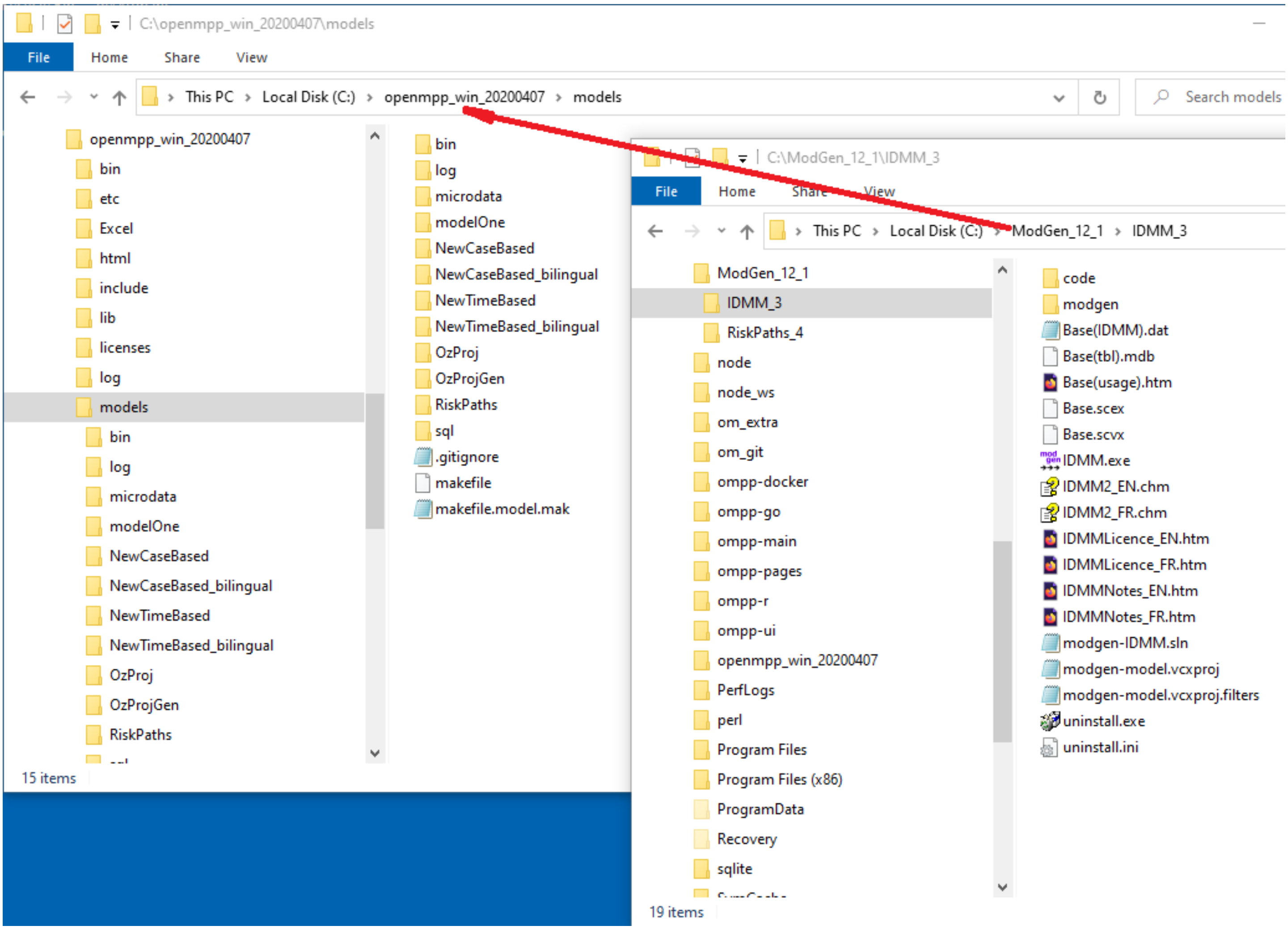
Modgen: Convert time-based model to openM++

Overview

OpenM++ provides superset of Modgen language specification and therefore able to compile Modgen source files. Conversion from Modgen include following:

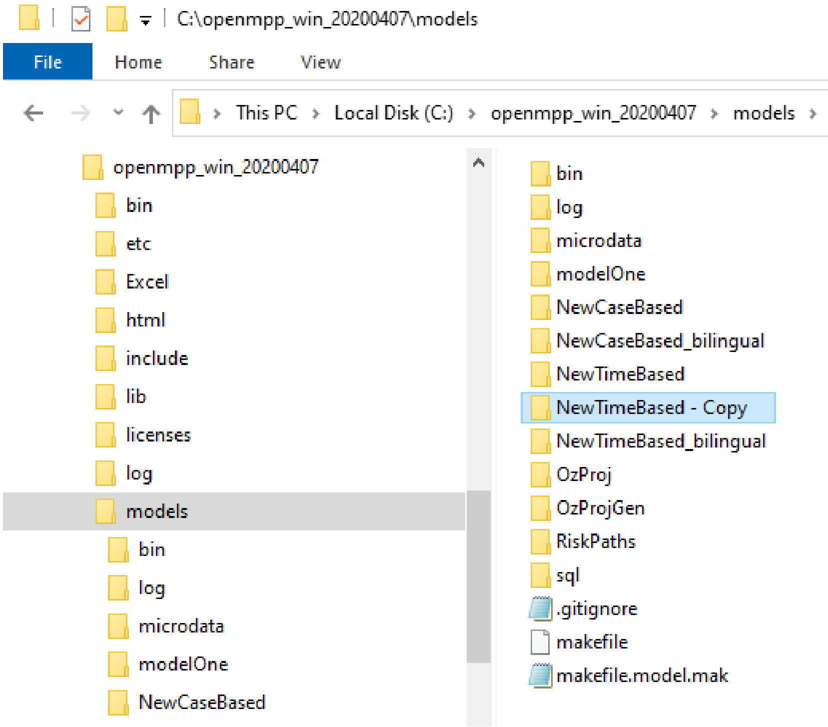
- Make sure you are done with: [Windows: Quick Start for Model Developers]
- Clone existing openM++ time-based model, for example: NewTimeBased
- Rename model directory and solution to YourModelName, for example: IDMM
- Replace NewTimeBased *.mpp modules with your model IDMM* .mpp files and inspect your code for any quirks (often none)
- Replace NewTimeBased *.dat parameter data with your model IDMM* .dat files
- Open Visual Studio, build the model and fix errors if necessary
- Run the model and verify simulation results

Below is step-by-step example how to convert IDMM model from Modgen 12.1 to openM++.



Clone existing openM++ model

As starting point please copy one of openM++ sample models, for time-based model we can start from NewTimeBased.

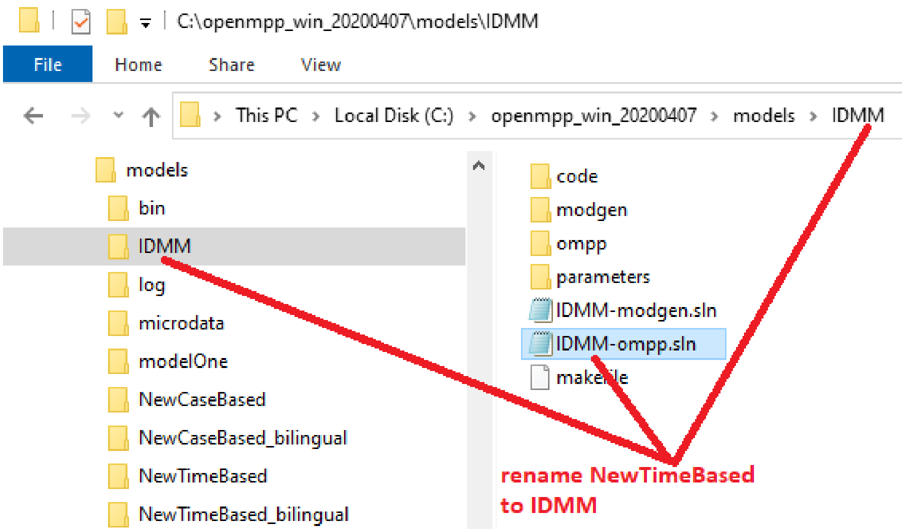


Rename model directory and solution

Rename directory and model solution into `YourModelName.sln`:

- rename `NewTimeBased - Copy` directory into `IDMM`
- rename `NewTimeBased-ompp.sln` into `IDMM-ompp.sln`
- (optional) rename `NewTimeBased-modgen.sln` into `IDMM-modgen.sln`

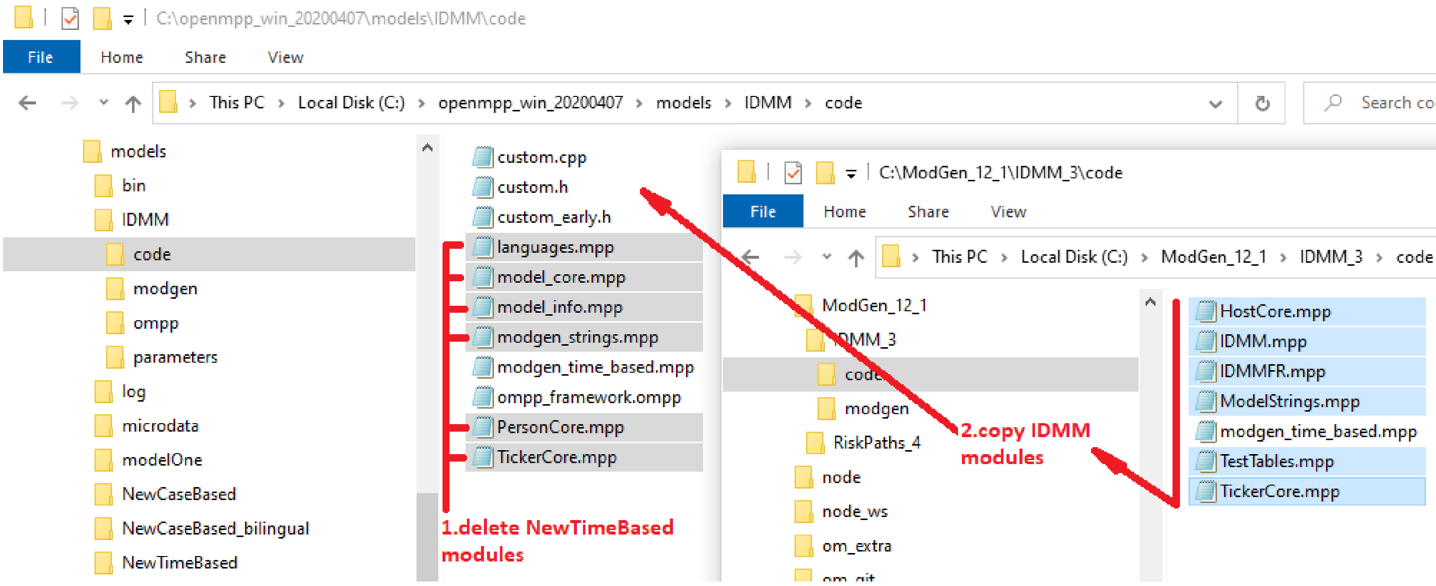
Note: It is not required to use model name as directory name and solution name, but it is openM++ convention and significantly simplifies model maintenance.

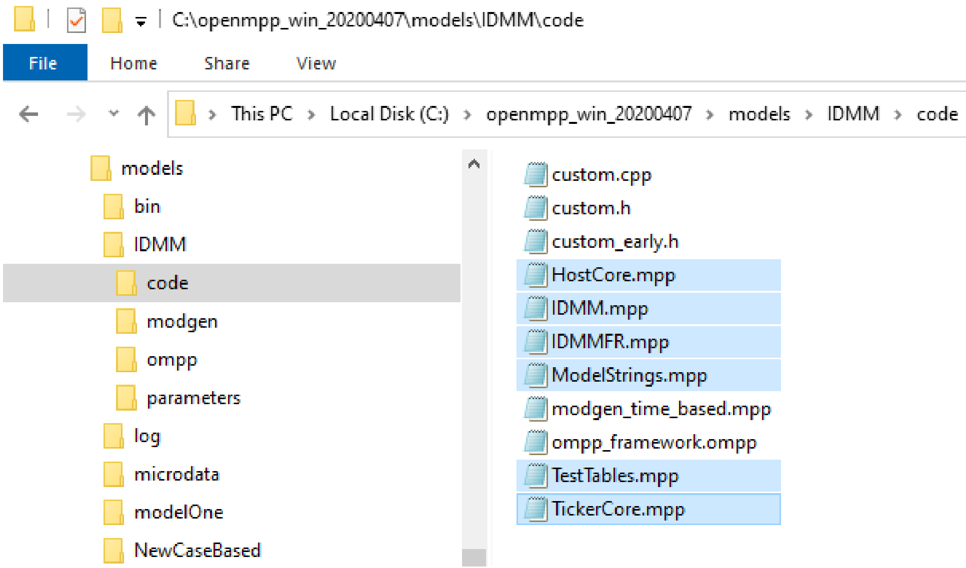


Replace sample model *.mpp modules with your model* .mpp files

Delete NewTimeBased *.mpp modules and copy your model substantive* .mpp files instead. For complex models with long maintenance history it may be not always easy to understand what *.mpp files are "substantive" and you may need to repeat this step multiple times.

It is also rare, but possible for some *.mpp modules to contain special quirky code in order to overcome issues in old version of Modgen or c++. Please inspect your code and adjust it, if necessary, to comply with c++17 standard.



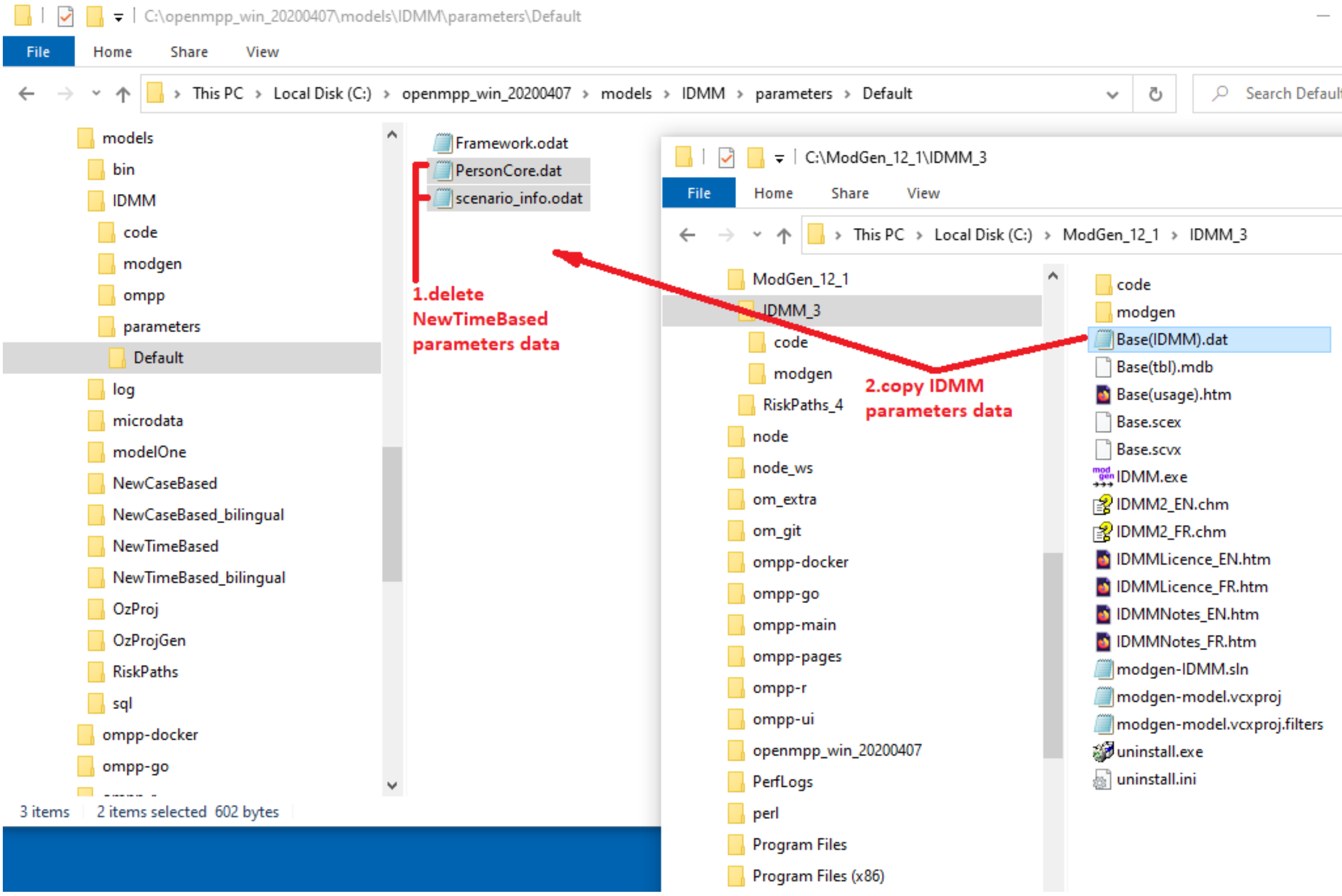


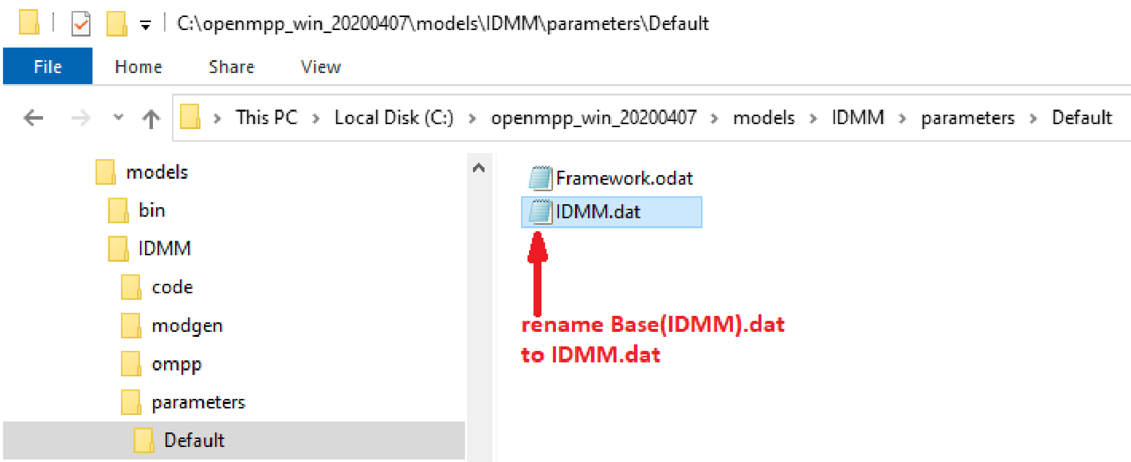
Replace sample model parameter data with your model *.dat files

For our example we need to:

- delete NewTimeBased `parameters/Default/PersonCore.dat` and `parameters/Default/scenario_info.odat`
- copy `Base(IDMM).dat`
- (optional) rename it into `IDMM.dat`

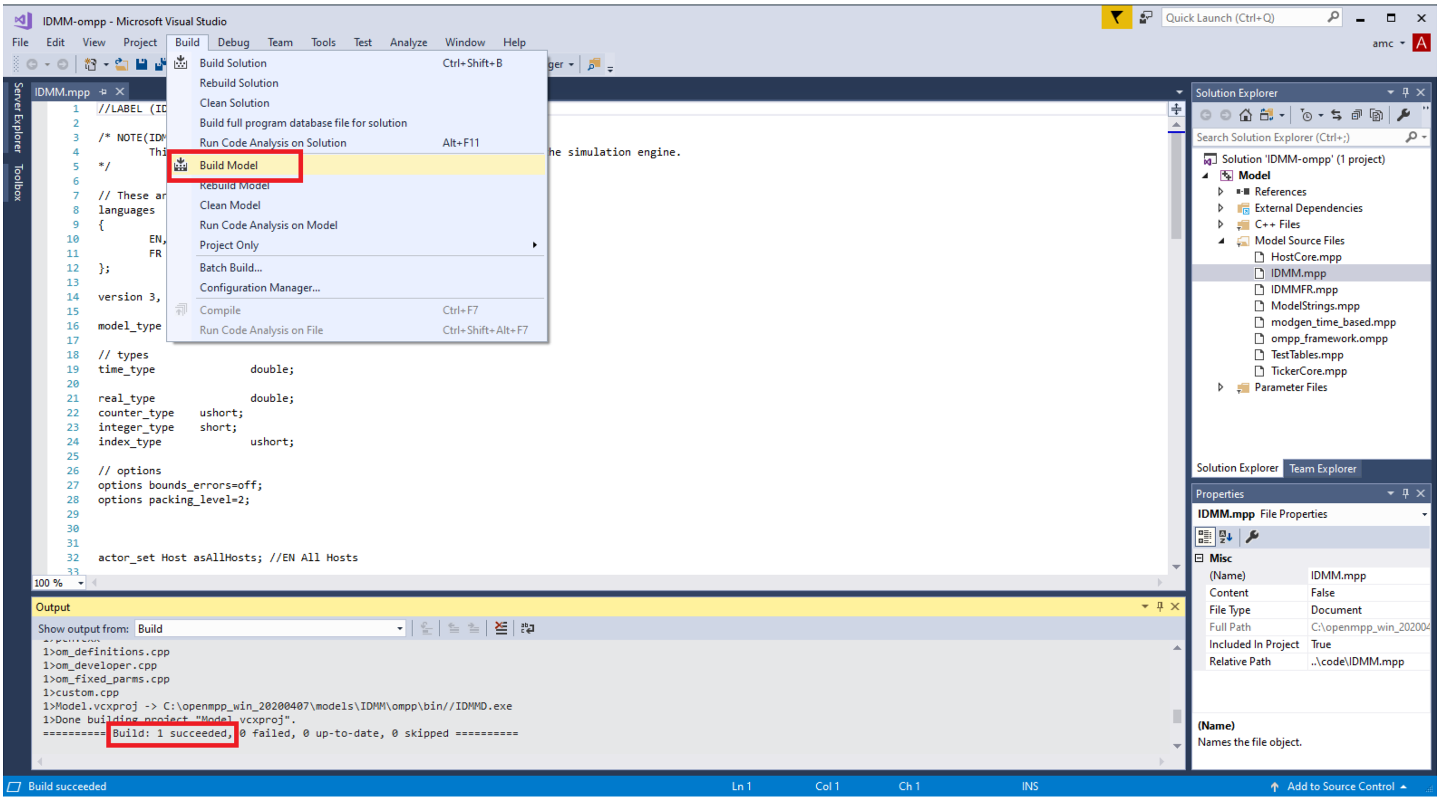
For complex models it is also possible to have `Fixed` parameters data. Please copy it into `parameters/Fixed/` sub-folder.





Open Visual Studio solution and build the model

Open `IDMM-ompp.sln` solution in Visual Studio and build the model, fix errors, if necessary.



Run the model and verify simulation results

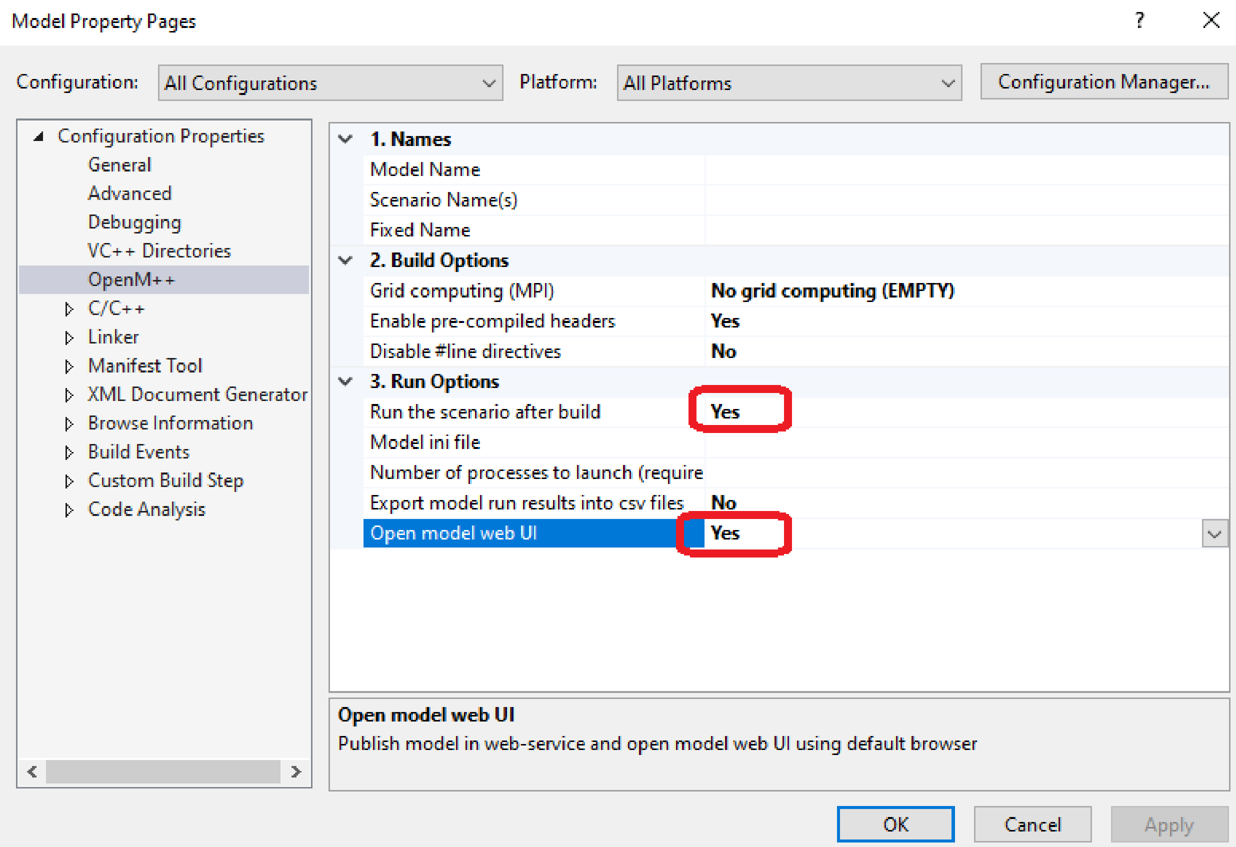
Last, but obviously very important step, is to run the model and compare Modgen and openM++ simulation results.

Check `parameters/Default/Framework.odat` values:

```
parameters {
  int SimulationSeed = 16807;
  Time SimulationEnd = 101.0;
};
```

and adjust number of simulation end time if required, re-build the model to update `SimulationEnd` value in `IDMM.sqlite` model database.

You can run openM++ model from command line, or from Visual Studio by changing `Project -> Properties -> OpenM++ -> Run Options`:



It is possible to open model run results in openM++ UI (beta version) to examine model parameters and output results:

Modgen: Convert Modgen models and usage of C++ in openM++ code

This page is under construction

Microdata files, gpoEventQueue, StartCase(), SignalCase()

It may be your model is using microdata files or it contains references to Modgen global variables: `gpoEventQueue, gbCanceled, gbErrors` or functions: `StartCase(), SignalCase()`. For example if your Modgen code look like:

```
// The Simulation function is called by Modgen to simulate a set of cases.
void Simulation()
{
  // Open the microdata file
  PersonOpenFile();

  // The global variables gbInterrupted, gbCancelled and gbErrors
  // are maintained by the Modgen run-time.
  for (long lCase = 0; lCase < CASES() && !gbInterrupted && !gbCancelled && !gbErrors; lCase++ )
  {
    // Simulate a case.
    // Tell the Modgen run-time to prepare to simulate a new case.
    StartCase();

    // Read the record corresponding to the case_id of the case
    long lCaseID = GetCaseID();
    PersonGetRecord(lCaseID);

    // Call the CaseSimulation function defined earlier in this module.
    CaseSimulation();

    // Tell the Modgen run-time that the case has been completed.
    SignalCase();
  }

  // Close the microdata file
  PersonCloseFile();
}

// The CaseSimulation function simulates a single case
void CaseSimulation( )
{
    // ..................
    // Process events until there are no more
    ProcessEvents();
    // ..................
}

// The ProcessEvents function processes all events until there are none in the event queue.
// It is called by the CaseSimulation function.
void ProcessEvents()

  // The Modgen run-time implements the global event queue gpoEventQueue.
  while ( !gpoEventQueue->Empty() )
  {
    // The global variables gbCancelled and gbErrors
    // are maintained by the Modgen run-time.
    if ( gbCancelled || gbErrors )
    {
      // The user cancelled the simulation, or run-time errors occurred.
      // Terminate the case immediately.
      gpoEventQueue->FinishAllActors();
    }
    else
    {
      // Age all actors to the time of the next event.
      gpoEventQueue->WaitUntil( gpoEventQueue->NextEvent() );

      // Implement the next event.
      gpoEventQueue->Implement();
    }
  }
}
```

Then please use OzProj example model to update your CaseSimulation() function. There are Modgen version of `code_original\OzProj.mpp` and openM++ version: `code\OzProj.mpp` which you can use as starting point to upgrade your model code.

Use of ternary operator may require cast to underlying type

Use of ternary operator may require cast to underlying type (type name followed by `_t`). The Microsoft VC++ error number is a strong hint. The error message text is not helpful.

Assignments from one attribute to another may require cast to underlying type.

Assignments from one attribute to another may require cast to underlying type. Specific Microsoft VC++ error number helps to indicate the occurrence (the error message text is not helpful).

Use of min and max may need to be changed to specify the underlying type.

Use of min and max may need to be changed to specify the underlying type. We would recommend to invoke the template explicitly, eg `std::max<double>(a, b)`

Arguments to print-style functions need to be cast to explicit types.

Non-standard Microsoft functions and types must be replaced with standard.

Non-standard Microsoft functions and types must be replaced with standard. It is easy to detect such error: build your model on MacOS or Linux and to detect all non-standard Microsoft extensions.

Model Localization: Translation of model messages

Home > Model Development Topics > **Model Localization**

This topic describes how to provide translations for model-specific run-time messages.

Related topics

- Multilingual Support **forthcoming content**

Topic contents

- Quick start
- How model finds translated message
- Model developer: How to mark strings for translation in model code

Quick Start

You can provide translated messages for your model by editing `modelName.message.ini` file located in the same directory where `modelName.exe` is.

For example:

```
dir /B openmpp_win_20180205\models\bin
....
modelOne.exe
modelOne.ini
modelOne.message.ini
modelOne.sqlite
```

`modelOne.message.ini` is translated messages for `modelOne.exe`

Message.ini file **must be UTF-8 encoded** and it contain model translated messages:

```
;
; modelOne localized messages
;

[FR]
Run %d = Exécution: %d

[fr-CA]
Run %d = Exécution: %d
;
; Example of multi-line translated message:
;
"Scenario processing" = "\
  Traitment \
  du scénario\
  "

[en]
; Model = Model
```

If translation the same as original message you can exclude it, e.g.: `Model = Model` is not required.

[back to topic contents]

How model finds translated message

At start model determine list user preferred languages. For example if current environment is French Canadian and model default language is EN then language list will be: `(fr-ca, fr, en)`.

User language preference can be changed in Windows Control Panel or by Linux LANG environment variable. You can override environment language by using model command-line or ini-file argument:

```
modelOne.exe -OpenM.MessageLanguage es-EC
```

To find translated message model does lookup in:

- `modelName.message.ini`
- database table `model_word`
- database table `lang_word` Search done in order of user preferred languages.

For example, if `modelOne.message.ini` is same as above and database table `model_word` contains entry:

```
fr-CA  Done.  Fini.
```

Then model messages in French Canadian environment can be similar to:

```
2014-03-17 17:14:24.0023 Model: modelOne
2014-03-17 17:14:24.0070 Exécution 101
....
2014-03-17 17:14:24.0179 Fini.
```

As you can see for current user language `fr-CA` model found two messages translated in "generic `fr`" French: "Exécution" and "Fini", however "Model" still untranslated. To fix this you can update `modelOne.message.ini` by adding:

```
[fr-CA]
Model = Modèle
```

Then result would look like:

```
2014-03-17 17:14:24.0023 Modèle: modelOne
2014-03-17 17:14:24.0070 Exécution 101
....
2014-03-17 17:14:24.0179 Fini.
```

[back to topic contents]

Model developer: How to mark strings for translation in model code

Omc model compiler automatically include first `"const char *"` argument of

- `theLog->logMsg("some message");`
- `theLog->logFormatted("some format %d %g %s",....);`
- macro `LT("some message")`
- `WriteLogEntry("some message");`
- `WriteDebugLogEntry("some message");`
- `WarningMsg("some message");`
- `ModelExit("some message");` into output `model.message.ini` file, which can be used as translation starting point.

If your source code directory already contains translated `code/model.message.ini` then such file is merged with newly added model messages into output `bin/model.message.ini`, which you can forward to translation team.

It is possible to use macro `LT("something")` in order to build concatenated message, however LT is "strictly inline" because it returns temporary `const char *` pointer. As result following will crash your model:

```
const char * myBadDay = LT("nice day");
if (myBadDay .... // memory access violation, model crash
```

How to avoid string concatenation. String concatenation considered as bad practice by any translation guide. For example, if you have something like:

```
string msg = LT("Table has ") + std::to_string(rowCount) + LT(" rows");
theLog->logMsg(msg.c_str());
```

then try to replace it with:

```
theLog->logFormatted("Table has %d rows", rowCount);
```

Non-translatable strings. Not every output in your model you want to translate For example, you may don't want to translate your model trace output:

```
WriteDebugLogEntry(NO_LT("------------"));
WriteDebugLogEntry(NO_LT("{1, 2, 3, 4}"));
WriteDebugLogEntry(NO_LT("------------"));
```

Please use `NO_LT` macro to disable unnecessary translation.

[back to topic contents]

How To: Set Model Parameters and Get Results

Overview

There multiple examples how to set input parameters, run the model and get results:

- run model from Python: simple loop over model parameter
- run RiskPaths model from Python: advanced parameters scaling
- run model from R: simple loop over model parameter
- run model from R: simple loop in cloud
- run RiskPaths model: advanced parameters scaling
- run RiskPaths model from R: advanced run in cloud
- oms web-service: How to prepare model input parameters

Also openM++ support following APIs:

- oms: openM++ web-service which you can use from any modern environment: Python, .NET, JavaScript, etc.
- openMpp R package
- Go library and tools

Quick examples below do not cover all possible options, please check links above and Model Run: How model finds input parameters for more details.

Sub-values: sub-samples, members, replicas

Following terms: "simulation member", "replica", "sub-sample" are often used in micro-simulation conversations interchangeably, depending on context. To avoid terminology discussion openM++ uses "sub-value" as equivalent of all above and some older pages of that wiki may contain "sub-sample" in that case.

Model output tables: sub-values, accumulators and expressions

There are two kind of model output tables:

- accumulators table: output sub-values (similar to Modgen sub-samples)
- expressions table: model output value calculated as accumulators aggregated across sub-values (e.g. `mean` or `CV` or `SE`)

All output accumulator tables always contain same number of sub-values, for example model run:

```
model.exe -OpenM.SubValues 16
```

will create 16 sub-values for each accumulator in each output accumulator table.

Model parameters: sub-values (optional)

OpenM++ parameters can also contain sub-values. Parameters sub-values are not required, it is a user choice to run the model and supply sub-values for some parameters.

For example, if user wants to describe statistical uncertanty of parameter `SalaryByYearByProvince` then csv file with 16 sub-values can be supplied tc run the model:

```
model.exe -OpenM.SubValues 16 -SubFrom.SalaryByYearByProvince csv -OpenM.ParamDir C:\MyCsv\
```

Parameters: Re-use same parameters values as in previous model run

Most of the model parameters are not changing between simulations and only few are varying. It is convenient to select all unchanged parameters from previous model run (from "base" run):

```
model.exe -Parameter.Ratio 0.7 -OpenM.BaseRunId 1234
model.exe -Parameter.Ratio 0.7 -OpenM.BaseRunDigest 5dc848891ea57db19d8dc08ec7a30804
model.exe -Parameter.Ratio 0.7 -OpenM.BaseRunName "My base run of the Model"
```

Above command do run the model with parameter `Ratio = 0.7` and the rest of parameters values are the same as it was in previous run with `id = 1234`.

It is also possible to use run diegst or run name to identify "base" model run:

```
model.exe -Parameter.Ratio 0.7 -OpenM.BaseRunDigest 5dc848891ea57db19d8dc08ec7a30804
model.exe -Parameter.Ratio 0.7 -OpenM.BaseRunName "My base run of the Model"
```

Please keep in mind, model run may not be unique and if database contains multiple model runs with the same name then first run will be selected.

Parameter: Value as command line argument

It is possible to specify value of any scalar parameter as command line argument, for example:

```
model.exe -Parameter.Ratio 0.7
```

There is an example of such technique at [Run model from R: simple loop over model parameter](#) page, where we using NewCaseBased model to study effect of Mortality Hazard input parameter on Duration of Life output:

```
for (mortalityValue from 0.014 to 0.109 by step 0.005)
{
  # run the model
  NewCaseBased.exe -Parameter.MortalityHazard mortalityValue
}
```

If parameter is enum-based (e.g. classification) then you can specify code or enum id:

```
modelOne.exe -Parameter.baseSalary Full
modelOne.exe -Parameter.baseSalary 22 -OpenM.IdParameterValue true
```

Parameter: Sub-values [0, N-1] as command line argument

If we want to run the model with multiple sub-values (a.k.a. sub-samples) and want "Grade" parameter sub-values to be created as [0, N-1] then:

```
model.exe -OpenM.SubValues 16 -SubFrom.Grade iota
```

as result sub-values parameter `Grade` would be: [0, ..., 15]

Parameter: Value inside of ini.file

Also any scalar parameter can be defined in model ini-file, i.e.:

```
model.exe -ini my.ini
```

```
; inside of my.ini file:
;
[Parameter]
Z_Parameter = XYZ      ; string parameter
SomeInt     = 1234     ; integer parameter
OrLogical   = true     ; boolean parameter
Anumber     = 9.876e5  ; float parameter
```

Parameters: Csv files

It is also possible to supply some (or even all) model parameters as csv-file(s). For example:

```
model.exe -OpenM.ParamDir C:\my_csv
```

If directory `C:\my_csv\` exist and contains `parameterName.csv` files then model will use it parameter values.

It is important to describe your parameter values to make sure model users clearly understand scenario data. In order to do that you can supply `parameterName.LANG-CODE.md` file(s).

For example, `C:\my_csv\Sex.csv` values of "Sex" parameter:

```
sub_id,dim0,param_value
0,    F,   true
0,    M,   false
```

And parameter value notes `C:\my_csv\Sex.EN.md`:

```
Sex parameter values in this scenario contain indicators of increased gender-specific hazards.
```

Note: *As it is today Markdown content of parameter value notes may not always display correctly in openM++ UI.*

Parameters: Csv files with multiple sub-values

If user want to supply up to 32 sub-values of "Sex" parameter:

```
sub_id,dim0,param_value
0,    F,   true
0,    M,   false
1,    F,   true
1,    M,   true
.................
31,    F,   false
31,    M,   true
```

Important: Presence of multiple sub-values in csv file (or in database) does not mean model will use all parameter sub-values. Only explicitly specified parameter(s) receiving sub-values.

For example, if user run the model 3 times:

```
model.exe -OpenM.SubValues 16
model.exe -OpenM.SubValues 16 -OpenM.ParamDir C:\my_csv
model.exe -OpenM.SubValues 16 -OpenM.ParamDir C:\my_csv -SubFrom.Sex csv
```

- "Sex" parameter expected to be in database and no sub-values used
- "Sex" parameter value is sub-value 0 from `C:\my_csv\Sex.csv`
- "Sex" parameter using sub-values [0, 15] from `C:\my_csv\Sex.csv`

Output Tables: Suppress output tables

By default model calculate all output tables and write it into database as model run results. Sometime it may be convenient to save only some output tables to reduce a time of each model run. This can be done by either suppressing model output table(s) or table group(s):

```
model.exe -Tables.Suppress ageSexIncome
model.exe -Tables.Suppress ageSexIncome,fullAgeSalary,A_TablesGroup
```

Or by suppressing output for all tables except of some:

```
model.exe -Tables.Retain ageSexIncome
model.exe -Tables.Retain ageSexIncome,fullAgeSalary,A_TablesGroup
```

Suppress and Retain options are mutually exclusive and cannot be mixed. For example, this model run would fail:

```
model.exe -Tables.Suppress ageSexIncome -Tables.Retain fullAgeSalary
```

Use dbcopy: Export entire model into text files

```
dbcopy -m modelOne
dbcopy -m modelOne -dbcopy.Zip
dbcopy -m modelOne -dbcopy.NoAccumulatorsCsv
dbcopy -m modelOne -dbcopy.NoMicrodata
```

It will create modelOne directory and modelOne.Zip file with:

- all model metadata (e.g. parameters, description, notes,...) in .json files
- csv files with sets of model input parameters
- csv files with model run results, input parameters and microdata

Model run microdata can be huge and if you are not intersted in it then use `-dbcopy.NoMicrodata` to suppress it:

```
dbcopy -m modelOne -dbcopy.NoMicrodata
```

For each model run output table openM++ store expression values (e.g. average, CV, SE) and also accumulators. Accumulators are sub-samples (a.k.a. sub-values or members or replicas, etc.) which used to produces output table aggregated expression value(s). By default `dbcopy` do output both: output table expressions and accumulators. If you are interestred only expression values then use `-dbcopy.NoAccumulatorsCsv` to suppress accumulators and get your results faster:

```
dbcopy -m modelOne -dbcopy.NoAccumulatorsCsv
```

Use dbcopy: Export entire model into csv files

```
dbcopy -m modelOne -dbcopy.To csv
dbcopy -m modelOne -dbcopy.To csv -dbcopy.Zip
dbcopy -m modelOne -dbcopy.To csv -dbcopy.NoAccumulatorsCsv
```

It will create modelOne directory and modelOne.Zip file with:

- all model metadata (e.g. parameters, description, notes,...) in .csv files
- csv files with sets of model input parameters
- csv files with model run results and input parameters Each model run result and each input parameters set will be in separate sub-directory. Use `-dbcopy.NoAccumulatorsCsv` option to get your results faster by suppressing accumulators (a.k.a sub-sanples) output to CSV files.

Other variation of csv output is:

```
dbcopy -m modelOne -dbcopy.To csv-all
```

In that case all model runs will be in "all_model_runs" sub-directory and all input sets are in "all_input_sets".

Use dbcopy: Export set of input parameters into text files

```
dbcopy -m modelOne -s modelOne_other -dbcopy.ParamDir pDir
```

It will create `pDir` directory with:

- input parameters set metadata (name, description, notes,...) in .json file
- csv files with sets of model input parameters

Use dbcopy: Export model run results into text files

```
dbcopy -m modelOne -dbcopy.LastRun
dbcopy -m modelOne -dbcopy.RunId 101
dbcopy -m modelOne -dbcopy.RunName modelOne_2016_11_22_11_38_49_0945_101
dbcopy -m modelOne -dbcopy.LastRun -dbcopy.NoAccumulatorsCsv
```

It will create a directory with:

- model run metadata (name, description, notes,...) in .json file

- csv files with input parameters used to run the model
- csv files with model output tables values Use `-dbcopy.NoAccumulatorsCsv` option to get your results faster by suppressing accumulators (a.k.a sub-sanples) output to CSV files.

Use dbcopy: Import parameters from csv files into database

```
dbcopy -m myModel -s MyInput -dbcopy.ParamDir P -dbcopy.ToSqlite myModel.sqlite
```

If any `parameterName.csv` file(s) exist in directory `P` then it will be loaded into `MyInput` set of input parameters.

It is recommended to run `dbcopy -m modelOne -s modelOne_other -dbcopy.ParamDir P` to get familiar how csv files look like.

Use dbcopy: Import parameters, description and notes from text files into database

```
dbcopy -m myModel -s MyInput -dbcopy.ToSqlite myModel.sqlite
```

It will insert or update MyInput set of input parameters in database with:

- if json metadata file exist then input set description, notes and parameter value note updated
- if any parameterName.csv files exist then it will be loaded into database

It is recommended to run `dbcopy -m modelOne -s modelOne_other -dbcopy.ParamDir P` to get familiar how json and csv files look like.

Example of json metadata file for "ageSexData" input set of parameters with description, notes and `ageSex` parameter value notes:

```
{
  "ModelName" : "modelOne",
  "Name" : "ageSexData",
  "Txt" : [{
      "LangCode" : "EN",
      "Descr" : "Model One set of parameters"
    }
  ],
  "Param" : [{
      "Name" : "ageSex",
      "SubCount" : 1,
      "Txt" : [{
          "LangCode" : "EN",
          "Note" : "Age by Sex values"
        }
      ]
    }
  ]
}
```

It is also must exist csv file with parameter values: `ageSex.csv`

Example of json metadata file for "emptyData" input set of parameters with description and notes in English and French:

```
{
  "ModelName" : "modelOne",
  "Name" : "emptyData",
  "Txt" : [{
      "LangCode" : "EN",
      "Descr" : "Model One set of parameters",
      "Note" : "Notes for model One set of parameters"
    }, {
      "LangCode" : "FR",
      "Descr" : "Je suis désolé je ne parle pas français"
    }
  ]
}
```

Model Run: How model finds input parameters

Model run cycle overview

Model run (execution of the model) consists of the following steps:

- initializing of model process(es) with model run options
- connecting to database and creating "model run" with `run_id` and `run_name`
- find set of input parameters and prepare it for the run
- reading model input parameters
- simulation of sub-values
- writing output sub-values to output tables in database
- aggregating sub-values using Output Expressios

Results of model run stored in database within unique integer "run_id" and include all model parameters, options and output result tables. **You always can find full set of model input and output by run id.**

OpenM++ models can be run on Windows and Linux platforms, on single desktop computer, on multiple computers over network, in HPC cluster or cloud environment (Google Cloud, Microsoft Azure, Amazon,...). Because openM++ runtime library hides all that complexity from the model we can safely assume model is a single executable on local machine. Please check Model Run: How to Run the Model for more details.

Sub-values: sub-samples, members, replicas

Following terms: "simulation member", "replica", "sub-sample" are often used in micro-simulation conversations interchangeably, depending on context. To avoid terminology discussion openM++ uses "sub-value" as equivalent of all above and some older pages of our wiki may contain "sub-sample" in that case.

Model output tables: sub-values, accumulators and expressions

There are two kind of model output tables:

- accumulators table: output sub-values (similar to Modgen sub-samples)
- expressions table: model output value calculated as accumulators aggregated across sub-values (e.g. `mean` or `CV` or `SE`)

All output accumulator tables always contain same number of sub-values, for example model run:

```
model.exe -OpenM.SubValues 16
```

will create 16 sub-values for each accumulator in each output accumulator table.

Model parameters: sub-values (optional)

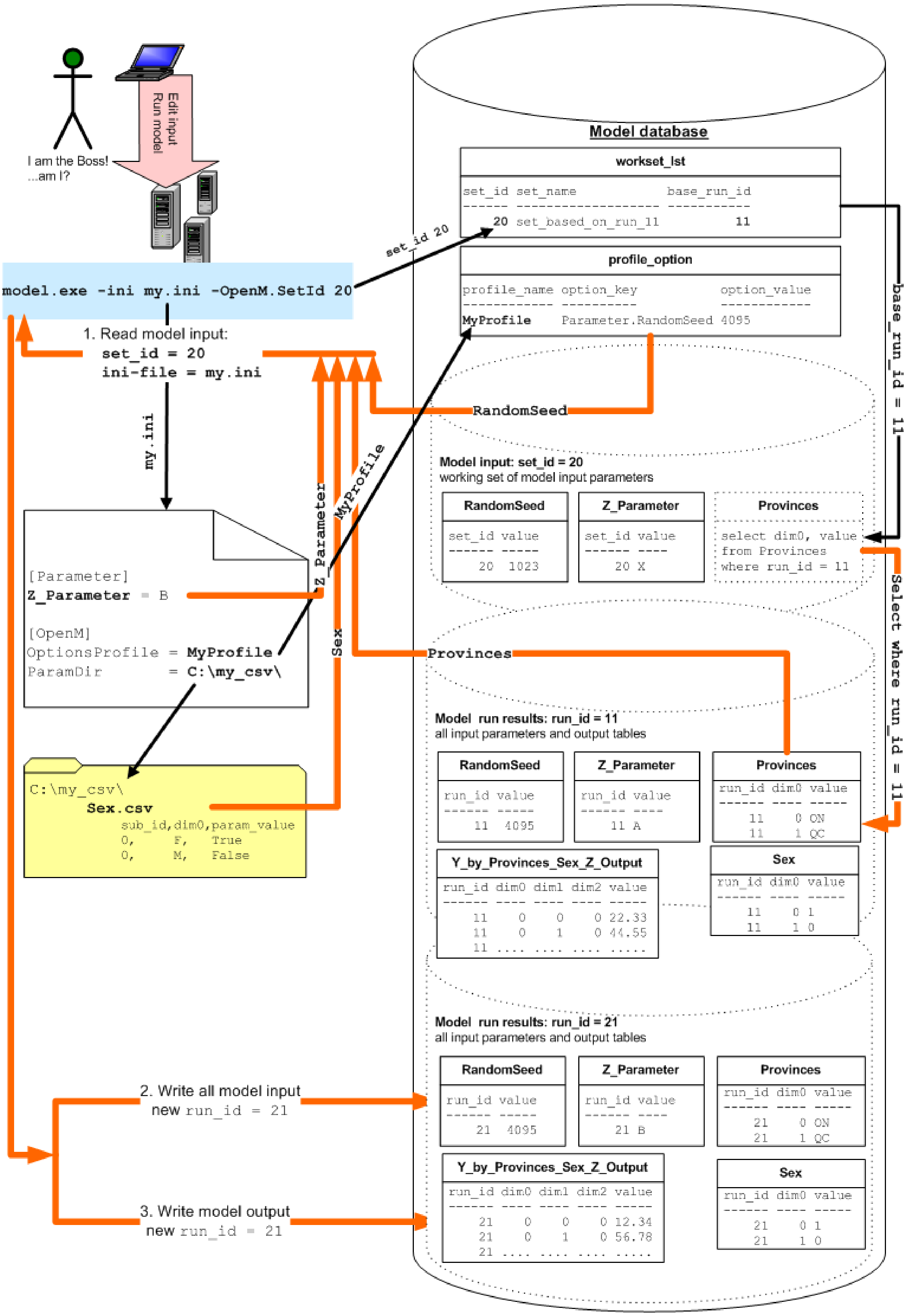
OpenM++ parameters can also contain sub-values. Parameters sub-values are not required, it is a user choice to run the model and supply sub-values for some parameters.

For example, if user wants to describe statistical uncertanty of parameter `SalaryByYearByProvince` then csv file with 16 sub-values can be supplied tc run the model:

```
model.exe -OpenM.SubValues16 SubFrom.SalaryByYearByProvince csv -OpenM.ParamDir C:\MyCsv\
```

Note: To simplify diagram below we do omit sub-values from the picture. But in real database there are multiple sub-values for parameters and accumulators; each sub-value identified by `sub_id` column.

How model finds input parameters: Parameters search order



Model search for input parameter values in following order:

- use parameter value specified as command line argument
- use parameter value specified inside of ini-file `[Parameter]` section
- use parameter value from profile_option table
- read parameter.csv file from "OpenM.ParamDir" directory

- import parameter value from other model parameter or other model output table
- use parameter value set of input parameters in database: workset
- use same value as in previous model run: values from "base" run
- use parameter value from default set of input parameters in database: default workset
- some parameters, e.g. number of sub-values may have default value

In any case all input parameters are copied under new run id before simulation starts. That process of copy parameters do guarantee a full copy of input parameters for each model run in database.

Model run options

There are many options which control model run, i.e.: number of sub-values, number of threads, etc. OpenM++ model gets run options in following order:

- as command line arguments
- from model run options ini-file
- from database `profile_option` tables
- use default values

Each option has unique key associated with it, e.g. "Parameter.RandomSeed" is model input parameter "RandomSeed", which is most likely, random generator starting seed. You can use this key to specify model parameter on command line, in ini-file or database. For example:

```
modelOne.exe -Parameter.RandomSeed 123 -ini my.ini
```

would run `modelOne` model with random seed = 123 and other options from `my.ini` file.

Please see OpenM++ Model Run Options to find out more.

Set of model input parameters in database (workset or scenario) and "base" model run

Database can contain multiple versions of model input parameter value. User can edit (change values of) input parameter(s) and save it as "working set of model input parameters" (a.k.a. "workset" or scenario).

- each set of parameters has unique "set id" and unique "set name"
- each model must have at least one full set of input parameters populated with default values (default set)
- default input set is a first set of model parameters (first means set with minimal set id)

Most of the model parameters are not changing between simulations and only few are varying. It is convenient to select all unchanged parameters from previous model run ("base" run). In order to do that user can:

- specify "base" model run to re-use parameters values
- create input set of parameters as "based on previous model run" and include only updated parameters in that input set Model will use parameters values from command line, csv files, etc. (as described above) and:
- if input set (workset) specified then select all parameters which do exist in that workset
- if "base" model run specified then select the rest parameters values from that previous model run
- if there is no "base" run then select model parameters from model default workset

How model finds input parameters: Default

If user run the model without any arguments:

```
modelOne.exe
```

then input parameters selected from default set, which is the first input data set of that model.

How model finds input parameters: Input set name or Id

To run the model with input data other than default user can specify set id or workset name:

```
modelOne.exe -OpenM.SetId 20
```

```
modelOne.exe -OpenM.SetName "My Set of Input Parameters"
```

assuming workset with `set_id = 20` and set with name `My Set of Input Parameters` exists in model database.

How model finds input parameters: re-use parameters from previous model run (base run)

It is often convenient to re-use parameters from previous model run:

```
model.exe -Parameter.Ratio 0.7 -OpenM.BaseRunId 42
```

As result model will be using same parameters values as it was for run with `run_id = 42` except of parameter `Ratio = 0.7`. For more details please see below: How to specify model base run.

How model finds input parameters: Value as command line argument

It is also possible to specify value of any scalar parameter as command line argument, i.e.:

```
model.exe -Parameter.Ratio 0.7
```

There is an example of such technique at Run model from R: simple loop over model parameter page, where we using NewCaseBased model to study effect of Mortality Hazard input parameter on Duration of Life output:

```
for (mortalityValue from 0.014 to 0.109 by step 0.005)
{
  # run the model
  NewCaseBased.exe -Parameter.MortalityHazard mortalityValue
}
```

How model finds input parameters: iota sub-values command line argument

If we want to run the model with N sub-values (a.k.a. sub-samples) and want `Grade` parameter sub-values to be created as [0,...,N-1] then:

```
model.exe -OpenM.SubValues 10 -SubFrom.Grade iota
```

as result sub-values of parameter `Grade` would be: [0, ..., 9]

How model finds input parameters: Value inside of ini.file

Also any scalar parameter can be defined in model ini-file, i.e.:

```
model.exe -ini my.ini
```

```
; inside of my.ini file:
;
[Parameter]
Z_Parameter = B        ; string parameter
SomeInt     = 1234     ; integer parameter
OrLogical   = true     ; boolean parameter
Anumber     = 9.876e5  ; float parameter
```

How model finds input parameters: Value in model profile

Another way to supply value of scalar parameter(s) is through `profile_option` database table. For example:

```
model.exe -OpenM.SetId 20 -OpenM.Profile MyProfile
```

```
SELECT * FROM profile_lst;

profile_name
------------
MyProfile

SELECT * FROM profile_option;

profile_name  option_key            option_value
------------- --------------------- ------------
MyProfile     Parameter.RandomSeed   4095
```

How model finds input parameters: Csv file

It is also possible to supply some (or even all) model parameters as csv-file(s). For example:

```
model.exe -OpenM.ParamDir C:\my_csv
```

If directory `C:\my_csv\` exist and contains `parameterName.csv` file model will use it parameter values. Parameter directory can be specified as command-line argument or as ini-file entry (it is not recommended to use `profile_option` table for `OpenM.ParamDir` option).

On picture above model run as:

```
model.exe -ini my.ini -OpenM.SetId 20
```

and my.ini file contains:

```
[OpenM]
ParamDir = C:\my_csv\
```

As result `model.exe` will read from `C:\my_csv\Sex.csv` values of "Sex" parameter:

```
sub_id,dim0,param_value
0,     F,   true
0,     M,   false
```

Together with csv files you can also supply parameter value note file(s) to describe scenario data values in each model language. Parameter value note files must be located in the same csv directory are named as: `parameterName.LANG-CODE.md`. For example, `C:\my_csv\Sex.EN.md` is an English notes for `Sex` parameter values:

```
Sex parameter values in this scenario contain indicators of increased gender-specific hazards.
```

It is also possible to have enum id's in csv files instead of codes, for example `C:\my_csv\Sex.csv` can be:

```
sub_id,dim0,param_value
0,     0,   true
0,     1,   false
```

To use such csv files you need to run the model with `OpenM.IdCsv true` argument:

```
model.exe -OpenM.SetId 20 OpenM.IdCsv true
```

Format of parameter.csv is based on RFC 4180 with some simplification:

- space-only lines silently ignored
- end of line can be CRLF or LF
- values are trimmed unless they are `" double quoted "`
- multi-line string values not supported

If parameter is boolean then following values expected (not case sensitive):

- "true" or "t" or "1"

- "false" or "f" or "0"

Important: Header line must include all dimension names, in ascending order, without spaces, e.g.: `sub_id,dim0,dim1,dim2,dim3,param_value`.

Parameter.csv file must contain all values, e.g. if parameter has 123456 values then csv must have all 123456 lines + header. Sorting order of lines are not important.

Csv file with multiple sub-values

If user want to supply up to 32 sub-values of "Sex" parameter then Sex.csv file look like:

```
sub_id,dim0,param_value
0,    F,   true
0,    M,   false
1,    F,   true
1,    M,   true
.................
31,    F,   false
31,    M,   true
```

Important: Presence of multiple sub-values in csv file (or in database) does not mean model will be using all parameter sub-values. Only explicitly specified parameter(s) receiving sub-values.

For example, if user run the model 8 times:

```
model.exe -OpenM.SubValues 8
model.exe -OpenM.SubValues 8 -OpenM.ParamDir C:\my_csv
model.exe -OpenM.SubValues 8 -OpenM.ParamDir C:\my_csv -SubFrom.Sex csv -SubValues.Sex default
model.exe -OpenM.SubValues 8 -OpenM.ParamDir C:\my_csv -SubFrom.Sex csv -SubValues.Sex 17
model.exe -OpenM.SubValues 8 -OpenM.ParamDir C:\my_csv -SubFrom.Sex csv
model.exe -OpenM.SubValues 8 -OpenM.ParamDir C:\my_csv -SubFrom.Sex csv -SubValues.Sex [24,31]
model.exe -OpenM.SubValues 8 -OpenM.ParamDir C:\my_csv -SubFrom.Sex csv -SubValues.Sex 1,3,5,7,9,11,13,15
model.exe -OpenM.SubValues 8 -OpenM.ParamDir C:\my_csv -SubFrom.Sex csv -SubValues.Sex xAAAA
model.exe -OpenM.SubValues 8 -OpenM.ParamDir C:\my_csv -SubFrom.GeoGroup csv -SubValues.GeoGroup 1,3,5,7,9,11,13,15
```

- "Sex" parameter expected to be in database and no sub-values used
- "Sex" parameter value is selected as "default" (sub_id=0) from `C:\my_csv\Sex.csv`, if .csv file exist
- "Sex" parameter value is selected as "default" (sub_id=0) from `C:\my_csv\Sex.csv`, .csv file must exist
- "Sex" parameter value is selected as sub_id = 17 from `C:\my_csv\Sex.csv`
- "Sex" parameter using sub-values [0,7] from `C:\my_csv\Sex.csv`
- "Sex" parameter using sub-values [24,31] from `C:\my_csv\Sex.csv`
- "Sex" parameter using sub-values 1,3,5,7,9,11,13,15 from `C:\my_csv\Sex.csv`
- "Sex" parameter using sub-values 1,3,5,7,9,11,13,15 from `C:\my_csv\Sex.csv` (bit mask)
- all parameters of GeoGroup using sub-values 1,3,5,7,9,11,13,15 from .csv files form `C:\my_csv\` directory

"Default" sub-value id can be explicitly defined for input parameter by person who published input set of parameters (workset). If "default" sub_id is not defined for that parameter then sub_id=0 assumed. Sub-value id's in the input set of parameters (in workset) can have be any integer (can be negative and not even have to sequential). For example if RatioByProvince parameter have 32 sub-values then typically sub_id's are [0,31], but it can be [-10, -8, -6, -4, -2, 0, 2, 4, ..., 52] and default sub_id can be = -10.

Important: Number of sub-values in csv must be at least as user required. In example above `Sex.csv` contains 32 sub-values and user cannot rur model with more than 32 sub-values.

How model finds input parameters: Import value from upstream model

If input parameter specified as "importable" by model developer then value(s) can be imported from run values of upstream model parameter or output table. For example if model developer of `BigModel` specified:

```
import Phi (RedModel.RedPhi) sample_dimension= off;
import Zet (SunModel.SunZet) sample_dimension= off;
```

And model user running `BigModel` as:

```
BigModel.exe -Import.All true
```

Then:

- value of `BigModel` parameter `Phi` must be imported from last run of `RedModel` parameter `RedPhi`
- value of `BigModel` parameter `Zet` must be imported from last run of `SunModel` output table `SunZet`

There are multiple options to control model import. For example if user run `BigModel` 9 times:

```
BigModel.exe -Import.All true
BigModel.exe -Import.SunModel true
BigModel.exe -ImportRunDigest.SunModel abcdefghef12345678
BigModel.exe -ImportRunId.SunModel 123
BigModel.exe -ImportRunName.SunModel GoodRun
BigModel.exe -ImportDigest.SunModel 87654321fedcba
BigModel.exe -ImportId.SunModel 456
BigModel.exe -ImportExpr.SunZet expr4
BigModel.exe -ImportDatabase.SunModel "Database=../NewSunModel.sqlite;OpenMode=ReadOnly;"
```

- Import all importable parameters from last successful run of upstream models
- Import all parameters importable from `SunModel` using values of last successful run of `SunModel`
- Import all parameters importable from `SunModel` using values of run where digest = `abcdefghef12345678`
- Import all parameters importable from `SunModel` using values of run where id = 123
- Import all parameters importable from `SunModel` using values of last successful run where run name = `GoodRun`
- Import all parameters importable from `SunModel` where model digest is `87654321fedcba` using values of last successful run
- Import all parameters importable from `SunModel` where model id = 456 using values of last successful run
- Import parameter `Zet` from `SunModel` output table `SunZet` expression `expr4` using values of last successful run
- Import all parameters importable from `SunModel` from database `../NewSunModel.sqlite`

Import options can be combined with sub-values options if model user want to select specific sub-values from upstream model parameter.

Default database to search for upstream model:

- if upstream model `SunModel` exist in current model database then it is imported from current database
- else it must be default upstream model SQLite database: `SunModel.sqlite`

How model finds input parameters: Value from previous model run (base run)

Most of the model parameters are not changing between simulations and only few parameters are varying. In that case it is convenient to select unchanged parameters from previous model run ("base" run).

Base run can be identified by `run_id` or run digest or run name. **Please note:** model run names are not unique and if there are multiple runs in database with the same name then first run selected:

```
SELECT MIN(run_id) WHERE run_name = 'Default model run';
```

Create set of input parameters based on previous model run

Input set of model parameters (workset) can be created as "based on existing run" and store only small number of model parameters, all the rest will be selected selected from "base" run by `run_id`.

On picture above command line to run the model is:

```
model.exe -ini my.ini -OpenM.SetId 20
```

and input set with id 20 defined as "based on run" with id = 11:

```
SELECT set_id, set_name, base_run_id FROM workset_lst WHERE set_id = 20;

set_id set_name           base_run_id
------ ------------------ -----------
    20 set_based_on_run_11     11
```

Because workset with id = 20 does not include "Provinces" input parameter those values selected from existing model run by `run_id = 11`:

```
SELECT dim0, param_value FROM Provinces WHERE run_id = 11;

dim0 value
---- -----
   0    ON
   1    QC
```

Note: sql above specially simplified, actual database table names, column names and queries bit more complex.

How to specify model base run

It is possible to explicitly specify model base run to select input parameters. For example:

```
model.exe -Parameter.Ratio 0.7 -OpenM.SetName "Age Input Values" -OpenM.BaseRunId 42
```

Model will use parameter `Ratio = 0.7` and select all parameters which do exist in `Age Input Values` workset:

```
SELECT dim0, param_value FROM Age WHERE set_name = 'Age Input Values';

dim0 value
---- -----
   0    [0,21]
   1    22+

.... select all other parameters where parameter exist in 'Age Input Values' ....
```

And the rest of model parameters selected from base run:

```
SELECT dim0, param_value FROM Provinces WHERE run_id = 42;

dim0 value
---- -----
   0    BC
   1    NS
```

It is also possible to use run diegst or run name to identify "base" model run:

```
model.exe -Parameter.Ratio 0.7 -OpenM.BaseRunDigest 5dc848891ea57db19d8dc08ec7a30804
model.exe -Parameter.Ratio 0.7 -OpenM.BaseRunName "My base run of the Model"
```

Please keep in mind, model run may not be unique and if database contains multiple model runs with the same name then first run will be selected.

Parameter sub-values from database

If we want to run the model with multiple sub-values (a.k.a. sub-samples) and want "RatioByProvince" parameter sub-values selected from database:

```
model.exe -OpenM.SubValues 8 -SubFrom.RatioByProvince db
```

Model will select "RatioByProvince" parameter sub-values from default workset or from base run, if there are no RatioByProvince parameter in default workset. Database **must** contain at least 8 sub-values for "RatioByProvince".

```
model.exe -OpenM.SubValues 8 -SubFrom.GeoGroup db
```

For GeoGroup of parameters model will select sub-values from default workset or from base run, if there are no such parameter in default workset. Database **must** contain at least 8 sub-values for all parameters of GeoGroup.

For example:

```
SELECT sub_id, dim0, param_value FROM RatioByProvince WHERE run_id = 11;

sub_id dim0 value
------ ---- -----
     0    0  1.00
     0    1  1.01
     1    0  1.02
     1    1  1.03
     2    0  1.04
     2    1  1.05
     ...........
     31   0  1.31
     31   1  1.32
```

In that case first 8 sub-values will be selected with `sub_id` between 0 and 7.

There are multiple options to specify which sub-values to select from database, for example:

```
model.exe -OpenM.SubValues 8
model.exe -OpenM.SubValues 8 -SubFrom.RatioByProvince db
model.exe -OpenM.SubValues 8 -SubFrom.RatioByProvince db -SubValues.Sex [24,31]
model.exe -OpenM.SubValues 8 -SubFrom.RatioByProvince db -SubValues.Sex 1,3,5,7,9,11,13,15
model.exe -OpenM.SubValues 8 -SubFrom.RatioByProvince db -SubValues.Sex xAAAA
model.exe -OpenM.SubValues 8 -SubFrom.RatioByProvince db -SubValues.Sex default
model.exe -OpenM.SubValues 8 -SubFrom.RatioByProvince db -SubValues.Sex 17
model.exe -OpenM.SubValues 8 -SubFrom.GeoGroup db -SubValues.GeoGroup 17
```

- "RatioByProvince" parameter expected to be in database and no sub-values used
- "RatioByProvince" parameter using sub-values [0,7] from database
- "RatioByProvince" parameter using sub-values [24,31] from database
- "RatioByProvince" parameter using sub-values 1,3,5,7,9,11,13,15 from database
- "RatioByProvince" parameter using sub-values 1,3,5,7,9,11,13,15 from database (bit mask)
- "RatioByProvince" parameter value is selected as "default" (sub_id=0) from database
- "RatioByProvince" parameter value is selected as sub_id = 17 from database
- all parameters of GeoGroup are selected as sub_id = 17 from database

"Default" sub-value id can be explicitly defined for input parameter by person who published input set of parameters (workset). If "default" sub_id is not defined for that parameter then sub_id=0 assumed. Sub-value id's in the input set of parameters (in workset) can have be any integer (can be negative and not even have to sequential). For example if RatioByProvince parameter have 32 sub-values then typically sub_id's are [0,31], but it can be [-10, -8, -6, -4, -2, 0, 2, 4, ..., 52] and default sub_id can be = -10.

On the other hand, in model run results sub_id is always [0,N-1] for run parameters and output tables. For example:

```
model.exe -OpenM.SubValues 8 -SubFrom.RatioByProvince db -SubValues.Sex [24,31]
```

"RatioByProvince" parameter in model run will have sub_id column values: [0,7].

Model Output Expressions

Sub-values: sub-samples, members, replicas

Following terms: "simulation member", "replica", "sub-sample" are often used in micro-simulation conversations interchangeably, depending on context. To avoid terminology discussion openM++ uses "sub-value" as equivalent of all above and some older pages of that wiki may contain "sub-sample" in that case.

Model output tables: sub-values, accumulators and expressions

There are two kind of model output tables:

- accumulators table: output sub-values (similar to Modgen sub-samples)
- expressions table: model output value calculated as accumulators aggregated across sub-values (e.g. `mean` or `CV` or `SE`)

All output accumulator tables always contain same number of sub-values, for example model run:

```
model.exe -OpenM.Subvalues 16
```

will create 16 sub-values for each accumulator in each output accumulator table.

Sub-values (accumulators) output tables

During the simulation OpenM++ model collect the results in "accumulators" and, at the end, write it into output accumulators table(s). Each output accumulator table contains results of model executions for all sub-values.

For example:

Model output table "Salary by Sex" has two accumulators and two dimensions:

- salary: 0 = "Low", 1 = "Medium", 2 = "High"
- sex: 0 = "Female", 1 = "Male"

If we run that model twice, first time with one sub-value and second with eight sub-values then output results may look like:

```
SELECT
  run_id, dim0, dim1, acc_id, sub_id, acc_value
FROM modelone_201208171604590148_a0_salarySex
ORDER BY 1, 2, 3, 4, 5;

run_id  dim0    dim1    acc_id  sub_id  acc_value
------  ------  ------  ------  ------  ----------
11      0       0       0       0       50.0
11      0       0       1       0       1.0
11      0       1       0       0       60.0
11      0       1       1       0       2.0
11      1       0       0       0       51.6
11      1       0       1       0       2.0
11      1       1       0       0       62.0
11      1       1       1       0       3.0
11      2       0       0       0       53.2
11      2       0       1       0       3.0
11      2       1       0       0       64.0
11      2       1       1       0       4.0
12      0       0       0       0       50.0
12      0       0       0       1       100.0
12      0       0       0       2       150.0
12      0       0       0       3       200.0
12      0       0       0       4       250.0
12      0       0       0       5       300.0
12      0       0       0       6       350.0
12      0       0       0       7       400.0
12      0       0       1       0       1.0
....more results....
12      2       1       1       7       11.0
```

Columns are:

- run_id: is unique run id for that model execution; all model input parameters and output results can be found by run_id;
- dim0: salary dimension items;

- dim1: sex dimension items;
- acc_id: zero-based accumulator number;
- sub_id: zero-based sub-value number;
- acc_value: accumulator value;

Accumulators are low level simulation results and useful mostly to analyze simulation model itself.

Aggregated output values

On top of accumulator values for each sub-value model can produce more meaningful output results by using OpenM++ output expressions, i.e.: median value across all sub-values. To do that model developer (or model user) can specify output aggregation expression, for example, median value is: `OM_AVG(acc0)`.

Each "value" output table can contain unlimited (reasonably unlimited) amount of aggregation expressions. Each expression must include aggregation function(s) with accumulators as argument(s) and, optionally, other arithmetic operators and basic SQL functions, such as `ABS` or `SQRT`.

Following OpenM++ sub-values aggregation functions are supported:

- `OM_COUNT(...expr...)` - count of values across all sub-values, `OM_COUNT(acc0)` result in SQL:

  ```
  COUNT(acc0)
  ```

- `OM_SUM(...expr...)` - sum of values across all sub-values, `OM_SUM(acc0)` result in SQL:

  ```
  SUM(acc0)
  ```

- `OM_AVG(...expr...)` - average value over sub-values, `OM_AVG(acc0)` result in SQL:

  ```
  AVG(acc0)
  ```

- `OM_MAX(...expr...)` - maximum value over all sub-values, `OM_MAX(acc0)` result in SQL:

  ```
  MAX(acc0)
  ```

- `OM_MIN(...expr...)` - minimal value over all sub-values, `OM_MIN(acc0)` result in SQL:

  ```
  MIN(acc0)
  ```

- `OM_VAR(...expr...)` - variance over sub-values, `OM_VAR(acc0)` result in SQL:

  ```
  SUM( (acc0 - AVG(acc0) * (acc0 - AVG(acc0) ) / ( COUNT(acc0) - 1 )
  ```

- `OM_SD(...)` - standard deviation:

  ```
  SQRT(OM_VAR(...expr...))
  ```

- `OM_SE(...expr...)` - standard error:

  ```
  SQRT(OM_VAR(...expr...) / COUNT(...expr...))
  ```

- `OM_CV(...expr...)` - coefficient of variation:

  ```
  100 * ( OM_SD(...expr...) / AVG(...expr...) )
  ```

There are also non-aggregation functions available:

- `OM_IF(...condition... THEN ...expr... ELSE ....other...)` - if `condition` is true then return `expr` else return `other` (else part is optional). `OM_IF(acc0 > 1.5 THEN` `acc0 ELSE 1.5)` result in SQL:

```
CASE WHEN acc0 > 1.5 THEN acc0 ELSE 1.5 END
```

- `OM_DIV_BY(...expr...)` - wrap expression to make it suitable for denominator:

```
CASE WHEN ABS(acc0) > 1.0e-37 THEN acc0 ELSE NULL END
```

If your expression include divide by operator then it is strongly recommended to wrap a denominator into `OM_DIV_BY()` function to prevent an error when divisor is zero or very small value. For example, if your expression is `acc1 / acc0` then use do `acc1 / OM_DIV_BY(acc0)`.

Aggregation expression can be more complex than a single function, for example: `OM_SUM(acc0) / OM_COUNT(acc0)` is equivalent of `OM_AVG(acc0)`. And `OM_SD(acc1)` can be written as:

```
SQRT(OM_SUM( (acc1 - OM_AVG(acc1) * (acc1 - OM_AVG(acc1) ) / ( OM_COUNT(acc1) - 1 ))
```

It is possible, as you can see, combine and nest aggregation functions in the expression.

It is important to understand:

- openM++ does aggregation across the sub-values, or other word, COUNT() is (almost) always number of sub-values.
- aggregation done by underlying SQL database, so, only non-NULL accumulator values are aggregated, so, COUNT() is number of non-NULL accumultor values across sub-values.
- accumulators always must be inside some aggregation function, i.e. this is an error: `acc0 + OM_SUM(acc1)` because `acc0` is not aggregated.

If you want to aggregate simulation results in your own way then it is always possible to combine openM++ and standard SQL functions in some custom expression. For example, if sub-values of your model is parts of large population then your may want to collect count and sum in separate accumulators and instead of `OM_AVG(...)` use custom median expression, like:

```
OM_SUM(acc0) / OM_SUM(acc1)
```

Also it is recommended to warp denominator part into `OM_DIV_BY()` function and result is:

```
OM_SUM(acc0) / OM_DIV_BY( OM_SUM(acc1) )
```

Examples of aggregation expressions

OpenM++ output table expressions translated into SQL aggregation queries. For example, if we have accumulator table:

```
CREATE TABLE out4_sub
(
  run_id    INT        NOT NULL,
  dim0      INT        NOT NULL,
  dim1      VARCHAR(8) NOT NULL,
  sub_id    INT        NOT NULL,
  acc0      FLOAT NULL,
  PRIMARY KEY (run_id, dim0, dim1, sub_id)
);

SELECT run_id, dim0, dim1, sub_id, acc0 FROM out4_sub ORDER BY run_id, dim0, dim1 DESC;

run_id  dim0      dim1     sub_id   acc0
------  ----      ----     ------   ----
2       10        M        0        1
2       10        M        1        2
2       10        M        2        3
2       10        M        3        4
2       10        F        0        1.5
2       10        F        1        2.5
2       10        F        2        3.5
2       10        F        3        4.5
2       20        M        0        10
2       20        M        1        20
2       20        M        2        30
2       20        M        3        40
2       20        F        0        10.5
2       20        F        1        20.5
2       20        F        2        30.5
2       20        F        3        40.5
3       10        M        0        5
3       10        M        1        6
3       10        F        0        7
3       10        F        1        8
3       20        M        0        50
3       20        M        1        60
3       20        F        0        70
3       20        F        1        80
```

Please, keep in mind: this is simplified example and in real openM++ database sub-value tables look like as described at the top of the article.

Then following results would be produced by openM++ aggregation functions:

Count, Average, Sum, Min and Max:

```
SELECT
  S.run_id, S.dim0, S.dim1,
  COUNT(S.acc0) AS "cnt",
  AVG(S.acc0)   AS "avg",
  SUM(S.acc0)   AS "sum",
  MIN(S.acc0)   AS "min",
  MAX(S.acc0)   AS "max"
FROM out4_sub S
GROUP BY S.run_id, S.dim0, S.dim1
ORDER BY S.run_id, S.dim0, S.dim1 DESC;

run_id dim0   dim1   cnt     avg      sum      min    max
------ ----   ----   ---     ---      ---      ---    ---
2      10     M      4       2.5      10       1      4
2      10     F      4       3        12       1.5    4.5
2      20     M      4       25       100      10     40
2      20     F      4       25.5     102      10.5   40.5
3      10     M      2       5.5      11       5      6
3      10     F      2       7.5      15       7      8
3      20     M      2       55       110      50     60
3      20     F      2       75       150      70     80
```

Count, Average and Variance:

```
SELECT
  S.run_id, S.dim0, S.dim1,
  COUNT(S.acc0) AS "cnt",
  AVG(S.acc0)   AS "avg",
  SUM(
    (S.acc0 - (SELECT AVG(VM1.acc0) FROM out4_sub VM1 WHERE VM1.run_id = S.run_id AND VM1.dim0 = S.dim0 AND VM1.dim1 = S.dim1) ) *
    (S.acc0 - (SELECT AVG(VM2.acc0) FROM out4_sub VM2 WHERE VM2.run_id = S.run_id AND VM2.dim0 = S.dim0 AND VM2.dim1 = S.dim1) )
  ) /
  ( (SELECT COUNT(VC1.acc0) FROM out4_sub VC1 WHERE VC1.run_id = S.run_id AND VC1.dim0 = S.dim0 AND VC1.dim1 = S.dim1) - 1) AS "var"
FROM out4_sub S
GROUP BY S.run_id, S.dim0, S.dim1
ORDER BY S.run_id, S.dim0, S.dim1 DESC;

run_id dim0     dim1    cnt     avg      var
------ ---      ----    ---     ---      ---
2       10       M       4       2.5      1.66666666666667
2       10       F       4       3        1.66666666666667
2       20       M       4       25       166.666666666667
2       20       F       4       25.5     166.666666666667
3       10       M       2       5.5      0.5
3       10       F       2       7.5      0.5
3       20       M       2       55       50
3       20       F       2       75       50
```

Count, Average and Standard Deviation:

```
SELECT
  S.run_id, S.dim0, S.dim1,
  COUNT(S.acc0) AS "cnt",
  AVG(S.acc0)   AS "avg",
  SQRT(
    SUM(
      (S.acc0 - (SELECT AVG(SDM1.acc0) FROM out4_sub SDM1 WHERE SDM1.run_id = S.run_id AND SDM1.dim0 = S.dim0 AND SDM1.dim1 = S.dim1) ) *
      (S.acc0 - (SELECT AVG(SDM2.acc0) FROM out4_sub SDM2 WHERE SDM2.run_id = S.run_id AND SDM2.dim0 = S.dim0 AND SDM2.dim1 = S.dim1) )
    ) /
    ( (SELECT COUNT(SDC1.acc0) FROM out4_sub SDC1 WHERE SDC1.run_id = S.run_id AND SDC1.dim0 = S.dim0 AND SDC1.dim1 = S.dim1) - 1)
  ) AS "sd"
FROM out4_sub S
GROUP BY S.run_id, S.dim0, S.dim1
ORDER BY S.run_id, S.dim0, S.dim1 DESC;

run_id dim0     dim1    cnt     avg     sd
------ ----     ---     ---     ---     --
2       10       M       4       2.5     1.29099444873581
2       10       F       4       3       1.29099444873581
2       20       M       4       25      12.9099444873581
2       20       F       4       25.5    12.9099444873581
3       10       M       2       5.5     0.707106781186548
3       10       F       2       7.5     0.707106781186548
3       20       M       2       55      7.07106781186548
3       20       F       2       75      7.07106781186548
```

Count, Average, and Standard Error:

```
SELECT
  S.run_id, S.dim0, S.dim1,
  COUNT(S.acc0) AS "cnt",
  AVG(S.acc0)   AS "avg",
  SQRT(
    SUM(
      (S.acc0 - (SELECT AVG(SEM1.acc0) FROM out4_sub SEM1 WHERE SEM1.run_id = S.run_id AND SEM1.dim0 = S.dim0 AND SEM1.dim1 = S.dim1) ) *
      (S.acc0 - (SELECT AVG(SEM2.acc0) FROM out4_sub SEM2 WHERE SEM2.run_id = S.run_id AND SEM2.dim0 = S.dim0 AND SEM2.dim1 = S.dim1) )
    ) /
    ( (SELECT COUNT(SEC1.acc0) FROM out4_sub SEC1 WHERE SEC1.run_id = S.run_id AND SEC1.dim0 = S.dim0 AND SEC1.dim1 = S.dim1) - 1) /
    (SELECT COUNT(SEC2.acc0) FROM out4_sub SEC2 WHERE SEC2.run_id = S.run_id AND SEC2.dim0 = S.dim0 AND SEC2.dim1 = S.dim1)
  ) AS "se"
FROM out4_sub S
GROUP BY S.run_id, S.dim0, S.dim1
ORDER BY S.run_id, S.dim0, S.dim1 DESC;

run_id dim0     dim1    cnt     avg     se
------ ----     ----    ---     ---     --
2       10       M       4       2.5     0.645497224367903
2       10       F       4       3       0.645497224367903
2       20       M       4       25      6.45497224367903
2       20       F       4       25.5    6.45497224367903
3       10       M       2       5.5     0.5
3       10       F       2       7.5     0.5
3       20       M       2       55      5
3       20       F       2       75      5
```

Count, Average, an Coefficient of Variation:

```
SELECT
  S.run_id, S.dim0, S.dim1,
  COUNT(S.acc0) AS "cnt",
  AVG(S.acc0)   AS "avg",
  100.0 * (
    SQRT(
      SUM(
        (S.acc0 - (SELECT AVG(CVM1.acc0) FROM out4_sub CVM1 WHERE CVM1.run_id = S.run_id AND CVM1.dim0 = S.dim0 AND CVM1.dim1 = S.dim1) ) *
        (S.acc0 - (SELECT AVG(CVM2.acc0) FROM out4_sub CVM2 WHERE CVM2.run_id = S.run_id AND CVM2.dim0 = S.dim0 AND CVM2.dim1 = S.dim1) )
      ) /
      ( (SELECT COUNT(CVC1.acc0) FROM out4_sub CVC1 WHERE CVC1.run_id = S.run_id AND CVC1.dim0 = S.dim0 AND CVC1.dim1 = S.dim1) - 1)
    ) /
    (SELECT AVG(CVM3.acc0) FROM out4_sub CVM3 WHERE CVM3.run_id = S.run_id AND CVM3.dim0 = S.dim0 AND CVM3.dim1 = S.dim1)
  ) AS "cv"
FROM out4_sub S
GROUP BY S.run_id, S.dim0, S.dim1
ORDER BY S.run_id, S.dim0, S.dim1 DESC;

run_id dim0     dim1    cnt     avg     cv
------ ----     ----    ---     ---     --
2       10       M       4       2.5    51.6397779494322
2       10       F       4       3      43.0331482911935
2       20       M       4       25     51.6397779494322
2       20       F       4       25.5   50.6272332837571
3       10       M       2       5.5    12.8564869306645
3       10       F       2       7.5    9.42809041582064
3       20       M       2       55     12.8564869306645
3       20       F       2       75     9.42809041582063
```

SQL implementation details

In the previous section we are using simplified representation of accumulator table and SQL dialect, which is not compatible across all vendors. Real SQL aggregation queries can be found in `expr_sql` column of `table_expr` metadata table. For example if source model expression is:

```
(OM_SUM(acc0) / OM_SUM(acc2))
```

then result look like:

```
SELECT
  M1.run_id, M1.dim0, (SUM(M1.acc_value) / SUM(L1A2.acc2)) AS expr1
FROM RiskPaths_201410071856440009_a2_T03_FertilityByAge M1
INNER JOIN
(
  SELECT run_id, dim0, sub_id, acc_value AS acc2
  FROM RiskPaths_201410071856440009_a2_T03_FertilityByAge
  WHERE acc_id = 2
) L1A2
ON (L1A2.run_id = M1.run_id AND L1A2.dim0 = M1.dim0 AND L1A2.sub_id = M1.sub_id)
WHERE M1.acc_id = 0
GROUP BY M1.run_id, M1.dim0
```

Model Run Options and ini-file

Overview

There are many options which control model run, i.e.: number of cases, random generator starting seed, etc. OpenM++ model gets run options in following order:

- as command line arguments
- from ini-file (similar to Modgen .sce file)
- from database `profile_option` tables
- use default values

Each option has unique key string associated with it, i.e. "Parameter.StartingSeed" is model input parameter "StartingSeed", which is most likely, random generator starting seed. You can use this key to specify model parameter on command line, in ini-file or database. For example:

```
modelOne.exe -Parameter.StartingSeed 123 -ini small.ini
```

would run "modelOne" model with starting seed = 123 and other options from `small.ini` file.

Note: We recommend to use normal Windows command line cmd.exe. If you are using Windows PowerShell then it may be necessary to put "quotes" around command line options, e.g:

```
modelOne.exe "-Parameter.StartingSeed" 123 "-ini" "small.ini"
```

OpenM++ database connection

If database connection string is not specified then model try to open SQLite database OM_MODEL_NAME.sqlite (i.e.: `modelOne.sqlite`) in current working directory. Default database connection string is:

```
Database=OM_MODEL_NAME.sqlite; Timeout=86400; OpenMode=ReadWrite;
```

Please notice, Linux file names are case sensitive and `modelOne.sqlite` is different from `ModelOne.sqlite`.

You can specify database connection string as command line argument, i.e.:

```
modelOne.exe -OpenM.Database "Database=C:\My Model\m1.sqlite; Timeout=86400; OpenMode=ReadWrite;"
```

Or, more convenient, by using ini-file

```
modelOne.exe -ini C:\MyModel\small.ini
```

Following parameters allowed for SQLite database connection:

- Database - (required) database file name or URI, file name can be empty
- Timeout - (optional) table lock "busy" timeout in seconds, default=0
- OpenMode - (optional) database file open mode: ReadOnly, ReadWrite, Create, default=ReadOnly
- DeleteExisting - (optional) if true then delete existing database file, default: false

Please notice: to run the model you need `OpenMode=ReadWrite`.

Model development options

Model developer can pass an arbitrary run option from ini-file and use it to debug model code. In order to do that model should be started with following command line arguments:

```
model.exe -ini some.ini -OpenM.IniAnyKey
```

Or any of equivalent formats:

```
model.exe -ini some.ini          -OpenM.IniAnyKey true
model.exe -OpenM.IniFile some.ini -OpenM.IniAnyKey true
model.exe -OpenM.IniFile some.ini -OpenM.IniAnyKey 1
model.exe -OpenM.IniFile some.ini -OpenM.IniAnyKey yes
```

Special boolean option `-OpenM.IniAnyKey true` allow to pass any key and values to model development code from ini-file.

For example, you can process following ini-file development options:

```
[MyTest]
ShowReport = yes             ; true if: "yes", "1", "true" or empty value, false if missing
ReportStyle = readable        ; string option
MinimumTime = 1234.56         ; double value, use as default: -inf
LineCount = 4321              ; integer option
EntityId = 1234567890123456789  ; long long integer
SelectedNames = e1,e2,e3       ; comma separated list of event names
```

by including code below into `ompp_framework.ompp`:

```
// process development model run options from model ini-file
void ProcessDevelopmentOptions(const IRunOptions * const i_options)
{
using namespace std;

    bool isShowReport = i_options->boolOption("MyTest.ShowReport");
    string rptStyle = i_options->strOption("MyTest.ReportStyle");
    double minTime = i_options->doubleOption("MyTest.MinimumTime", -numeric_limits<double>::infinity());
    int lineCount = i_options->intOption("MyTest.LineCount", 0);
    long long entityId = i_options->longOption("MyTest.EntityId", 0);

    // option is a list of comma separated names
    list<string> evtList = openm::splitCsv(i_options->strOption("MyTest.SelectedNames"));

    // if option is not specified at all
    if (!i_options->isOptionExist("MyTest.ShowReport")) {
        // do something
    }

    // get a copy of all model run options, including openM++ standard options
    vector<pair<string, string>> allOpts = i_options->allOptions();

    // each option is a pair of key and value
    for (const auto & opt : allOpts) {
    // string key = opt.first;
    // string value = opt.second;
    }
}
```

Important:

Model development options should not be used as model parameters and should not affect modeling results. It is strictly for debugging and development purpose. OpenM++ does not provide any guarantee about model development options.

OpenM++ ini-file run options

To specify name of ini-file you can use `-s` or `-ini` or `-OpenM.IniFile` command line option. Please see OpenM++ ini-file format to find out more about ini-file structure supported by openM++.

Example of model ini-file:

```
;#============================================================================
;#
;# model parameters
;# any scalar model parameter can be specified in [Parameter] section
;# or as command line argument or in profile_option table
;
[Parameter]

;# random seed value
;
; StartingSeed = 16807

;# base salary is classification parameter
;# using enum code "Full" to specify parameter value
;# if [OpenM]IdParameterValue=true (see below) then we must use baseSalary=22 instead
```

```
;
; baseSalary = Full

;#===============================================================================
;#
;# openM++ run options
;#
;# OpenM++ boolean options:
;#   True value is any of:  "yes", "1", "true" or empty value
;#   False value is any of: "no"   "0", "false"
;# Boolean values are not case sensitive, e.g.: "yes" == "YES" and it is a true value
;
[OpenM]

;# number of sub-values, default: 1
;
; SubValues = 16

;# max number of modeling threads, default: 1
;#
;# if number of sub-values per process < number of modeling threads then sub-values run sequentially.
;# if more threads specified then sub-values run in parallel.
;#
;# for example:
;#   model.exe -OpenM.SubValues 8
;#   model.exe -OpenM.SubValues 8 -OpenM.Threads 4
;#   mpiexec -n 2 model.exe -OpenM.SubValues 31 -OpenM.Threads 7
;
; Threads = 4

;# if NotOnRoot is true then do not use "root" process for modeling
;# default value: false
;# empty value:   true
;#
;# this option can be used only if multiple model.exe processes are running in parallel
;# otherwise it has no effect.
;#
;# for example:
;#   (a) mpiexec -n 4 model.exe -OpenM.SubValues 16
;#   (b) mpiexec -n 4 model.exe -OpenM.SubValues 16 -OpenM.NotOnRoot true
;# both commands above do launch four model.exe processes
;# but in second case only three children are doing modeling
;# and root process dedicated to run controlling activity
;
; NotOnRoot = false

;# database connection string
;#    default database name: model_name.sqlite
;
; Database = "Database=modelOne.sqlite; Timeout=86400; OpenMode=ReadWrite;"

;# name of model run results
;# if not specified then automatically generated
;
; RunName = my-default-scenario

;# set id is an id of input set of model parameters
;#
;# default: min(set id)
;
; SetId = 101

;# set name is name of input set to get model parameters
;# if set name specified then it used to find set of model input parameters
;# if SetId option specified then SetName is ignored
;
; SetName = Default

;# if specified then use parameters from base run instead of input set
;# find base run by run id
;
; BaseRunId = 1234

;# if specified then use parameters from base run instead of input set
;# if BaseRunId option NOT specified then find base run by run digest
;
; BaseRunDigest = 6866f742cabab735ced1577c56b23e93

;# if specified then use parameters from base run instead of input set
;# if BaseRunId and BaseRunDigest options are NOT specified then find base run by run name
;# run name is not unique and as result it will be a first model run with that name
;
; BaseRunName = My_Model_Run

;# run id to restart model run (i.e. after power failure)
;
; RestartRunId =
```

```
;# task id is an id of modeling task
;# if modeling task id specified then
;# model will run all input sets included into that modeling task
;
; TaskId = 1

;# task name is name of modeling task
;# if task name specified then it used to get task id
;# if task id specified then set name is ignored
;
; TaskName = taskOne

;# task run name is name of modeling task run
;# if not specified then automatically generated
;
; TaskRunName = run-first-task-with-16-sub-values

;# task "wait":
;# default value: false
;# empty value:   true
;#
;# allow to dynamically append new input data into modeling task
;# modeling task not completed automatically
;# it is waiting until some external script signal:
;#   UPDATE task_run_lst SET status = 'p' WHERE task_run_id = 1234;
;
; TaskWait = false

;# profile name to select run options from profile_option database table
;
; Profile = modelOne

;# convert to string format for float, double, long double, default: %.15g
;
; DoubleFormat = %.15g

;# path to parameters csv file(s) directory
;# if specified then for each parameter where exist param/dir/parameterName.csv
;# values from csv file are used to run the model
;
; ParamDir = ./csv

;# if true then parameter(s) csv file(s) contain enum id's, default: enum code
;# default value: false
;# empty value:   true
;
; IdCsv = false

;# value of scalar parameter(s) can be specified in [Parameter] section (see above)
;# or as command line argument -Parameter.Name of model.exe
;#
;# if IdParameterValue is true
;# then scalar parameter(s) value is enum id's, default: enum code
;# default value: false
;# empty value:   true
;
; IdParameterValue = false

;# if true then use sparse output to database, default: false
;# default value: false
;# empty value:   true
;
; SparseOutput = false

;# if use sparse and abs(value) <= SparseNullValue then value not stored
;# default = FLT_MIN
;
; SparseNullValue = 1.0E-37

;# if positive then used to report percent completed of simulation, default: 1
;
; ProgressPercent = 1

;# if positive then used to report simulation progress, default: 0
;# for case based models it is number of cases completed and must integer value
;# for time based models it is time passed from first event and must positive value, e.g.: 0.1
;
; ProgressStep = 1000

;# language to display output messages
;# default: set in Windows Control Panel or by Linux LANG
;
; MessageLanguage = en-CA

;# process run stamp, default: log file time stamp
;# use it to find model run(s) in run_lst table
;# or model task run in task_run_lst table
```

```
;
; RunStamp = 2012_08_17_16_04_59_148

;# log settings:
;# log can be enabled/disabled for 3 independent streams:
;#   console         - standard output
;#   "last run" file - log file with specified name, overwritten on every model run
;#   "stamped" file  - log file with unique name, created for every model run
;#
;# "stamped" name produced from "last run" name by adding time-stamp and/or pid-stamp, i.e.:
;#   modelOne.log => modelOne.2012_08_17_16_04_59_148.987654.log
;
; LogToConsole     = true       ; log to console, default: true
; LogToFile        = true       ; log to file, default: true
; LogToStampedFile = false      ; log to "stamped" file
; LogUseTimeStamp  = false      ; use time-stamp in log "stamped" file name
; LogUsePidStamp   = false      ; use pid-stamp in log "stamped" file name
; LogFilePath      = model.log  ; log file path, default = current/dir/modelExeName.log
; LogNoMsgTime     = false      ; if true then do not prefix log messages with date-time
; LogRank          = false      ; if true then prefix log messages with MPI process rank
; LogSql           = false      ; debug only: log sql statements

;# trace settings:
;# trace can be enabled/disabled for 3 independent streams:
;#   console         - cout stream
;#   "last run" file - trace file with specified name, overwritten on every model run
;#   "stamped" file  - trace file with unique name, created for every model run
;#
;# "stamped" name produced from "last run" name by adding time-stamp and/or pid-stamp, i.e.:
;#   trace.txt => trace.2012_08_17_16_04_59_148.987654.txt
;#
;# If trace to file is enabled
;# then existing "last run" trace file is overwritten even if model does not write anything to trace output
;
; TraceToConsole     = false      ; trace to console, default false
; TraceToFile        = false      ; trace to file
; TraceToStampedFile = false      ; trace to "stamped" file
; TraceFilePath      = trace.txt  ; trace file path, default: current/dir/modelExeName.trace.txt
; TraceUseTimeStamp  = false      ; use time-stamp in trace "stamped" file name
; TraceUsePidStamp   = false      ; use pid-stamp in trace "stamped" file name
; TraceNoMsgTime     = true       ; if true then do not prefix trace messages with date-time
; TraceRank          = false      ; if true then prefix trace messages with MPI process rank

;#=============================================================================
;#
;# language-specific options
;#

[EN]
;#
;# model run description in English
;
; RunDescription = model run with 50,000 cases

;#
;# path to file with model run notes in English
;
; RunNotesPath = run_notes-in-english.md

;#
;# run entity description in English
;
; Person--EntityDescription = base Person entities

;#
;# path to file with entity run notes in English
;
; Person--EntityNotesPath = entity-run_notes-in-english.md

[FR]
;#
;# model run description in French
;
; RunDescription = je suis désolé je ne parle pas français

;#
;# path to file with model run notes in French
;
; RunNotesPath = run_notes-fr-français.md

;#=============================================================================
;#
;# Ouput tables suppression.
;#
;# It can be in one of the two forms:
;#   Suppress = ATable,BTable,Group1,Group2
;# Or:
;#   Retain   = ATable,BTable,Group1,Group2
```

```
;#   Retain   = ATable,BTable,Group1,Group2
;#
;# Suppress and Retain options are mutually exclusive and cannot be mixed.
;# For example, this model run would fail:
;#   model.exe -Suppress.A -Retain.B

[Tables]
;#
;# Suppress output table "ageSexIncome"
;# and suppress group of output tables "AdditionalTables"
;
; Suppress = ageSexIncome,AdditionalTables

;# Or suppress all output tables
;# except of "ageSexIncome" table and tables included into "AdditionalTables" group:
;
; Retain   = ageSexIncome,AdditionalTables

;#============================================================================
;#
;# where to find sub-values for model parameter or group of parameters: db, csv, iota
;
[SubFrom]

;# where to find sub-values for parameter "Member"
;# "iota" means create parameter "Member" sub-values as 0,1,...[OpenM].SubValues-1
;
; Member = iota

;# where to find sub-values for "baseSalary" parameter
;# "db" means read sub-values from input set (read from model database)
;# modelOne default input set has 4 sub-values for "baseSalary" and "salaryFull"
;
; baseSalary = db

;# where to find sub-values for "salaryFull" parameter
;# "csv" means read all sub-values from parameter.csv file
;# by default only one sub-value read from csv file
;
; salaryFull = csv

;# sub-value for all members of "age_sex_parameters" group coming from .csv files:
;#
;# age_sex_parameters = csv
;#
;# it is the same as:
;#   -SubFrom.ageSex csv -SubFrom.salaryAge csv
;# because this group consist of: "ageSex" and "salaryAge"

;#============================================================================
;#
;# how many sub-values to select for parameter and which sub id to select
;# it is also can be applied to the parameters group
;#
;# SubValues option can be:
;#   range:       SubValues.Age [1,4]
;#   list of id's: SubValues.Age 2,1,4,3
;#   bit mask:     SubValues.Age x0F
;#   single id:    SubValues.Age 7
;#   default id:   SubValues.Age default
;#
;# if you running:
;#   model.exe -OpenM.SubValues 4 -SubFrom.Age csv
;# then Age.csv file must have at least 4 sub values with sub id's 0,1,2,3
;#
;# to use only one single sub-value either specify "default" id:
;#   model.exe -OpenM.SubValues 4 -SubFrom.Age db -SubValues.Age default
;# or explicit sub-value id:
;#   model.exe -OpenM.SubValues 4 -SubFrom.Age csv -SubValues.Age 7
;#
;# to select 4 sub-values use [first,last] range or comma-separated list or bit mask:
;#   model.exe -OpenM.SubValues 4 -SubFrom.Age csv -SubValues.Age [4,7]
;#   model.exe -OpenM.SubValues 4 -SubFrom.Age csv -SubValues.Age 4,5,6,7
;#   model.exe -OpenM.SubValues 4 -SubFrom.Age csv -SubValues.Age xF0
;#
[SubValues]

; baseSalary = default
; isOldAge   = 4,2,1,3

;# use sub-values 2 and 3 for all members of "age_sex_parameters" group:
;
; age_sex_parameters = 2,3
;
;# it is the same as:
;#   -SubValues.ageSex 2,3 -SubValues.salaryAge 2,3
;# because this group consist of: "ageSex" and "salaryAge"
```

```
;#=============================================================================
;#
;# import model parameters from other model(s)
;
[Import]

;# if "All" is true then import all parameters (all parameters which has import statement).
;# default value: false
;# empty value:   true
;
; All = true
;
;# for each upstream model last succesful run is used to import parameters
;#
;# if "ModelName" is true then import all parameters from upstream "ModelName".
;# default value: false
;# empty value:   true
;# Example:
;#   import parameters from last succesful run of upstream model "RedModel"
;
; RedModel = true

;#=============================================================================
;#
;# import model parameters from run specified by run digest
;#
[ImportRunDigest]

;# Example:
;# import parameters from upstream model "RedModel" where run digest = abcdefghef12345678
;
; RedModel = abcdefghef12345678

;#=============================================================================
;#
;# import model parameters from run specified by run id
;
[ImportRunId]

;# Example:
;# import parameters from upstream model "RedModel" where run id = 101
;
; RedModel = 101

;#=============================================================================
;#
;# import model parameters from last sucessful run with specified run name
;
[ImportRunName]

;# Example:
;# import parameters from last successful run of upstream model "RedModel" where run name = GoodRun
;
; RedModel = GoodRun

;#=============================================================================
;#
;# import model parameters from last sucessful run of model with specified digest
;
[ImportDigest]

;# Example:
;# import parameters from last successful run of upstream model "RedModel" where model digest = 87654321fedcba
;
; RedModel = 87654321fedcba

;#=============================================================================
;#
;# import model parameters from last sucessful run of model with specified id
;
[ImportId]

;# Example:
;# import parameters from last successful run of upstream model "RedModel" where model id = 123
;
; RedModel = 123

;#=============================================================================
;#
;# import model parameter from specified expression of output table
;
[ImportExpr]

;# If upstream output table has multiple measure values (multiple expressions)
;# the by default first expression of output table is used to import parameter value.
;# To override default measure name (expression name) can be explicitly specified.
;#
;# Example:
```

```
;# import parameter from AgeTable of upstream model "RedModel" using "expr2" value as parameter values
;
; AgeTable = expr2

;#============================================================================
;#
;# import model parameter from specified model database
;
[ImportDatabase]

;# By default upstream model imported from the same database as current (downstream) model
;# or, if not exist there then from defalut SQLite database with name ModelName.sqlite
;# Use connection string to override default database rule.
;#
;# Example:
;# import parameters from upstream model "RedModel" in database ../RedHot.sqlite
;
; RedModel = "Database=../RedHot.sqlite;OpenMode=RedaOnly;"

;#============================================================================
;#
;# model development options
;#
;# Are available for processing in model code only if model.exe started with command line options:
;#
;# model.exe -ini iniFileName.ini -OpenM.IniAnyKey
;#
;# Or:
;#
;# model.exe -ini iniFileName.ini -OpenM.IniAnyKey 1
;# model.exe -ini iniFileName.ini -OpenM.IniAnyKey yes
;# model.exe -ini iniFileName.ini -OpenM.IniAnyKey true
;#
;# OpenM++ boolean options:
;#   True value is any of:  "yes", "1", "true"  or empty value
;#   False value is any of: "no"   "0", "false"
;# Boolean values are not case sensitive, e.g.: "yes" == "YES" and it is a true value
;
; [EventTrace]
;
; ReportStyle = readable
; ShowScheduling = yes           ; true if: "yes", "1", "true" or empty value
; MinimumTime = 1234.56          ; double value, default: -inf
; MaximumTime =                  ; double value, default: +inf
; SelectedEntities =             ; comma separated list of integers
; SelectedEvents =  e1,e2,e3     ; comma separated list of event names
;
; [LargeOutput]
;
; incomeByYear   = true   ;   4824 * 4 expression cells
; incomeByLow    = true   ;  48240 * 4 expression cells
; incomeByMiddle = true   ; 144720 * 4 expression cells
; incomeByPeriod = true   ; 969624 * 4 expression cells
;

;#============================================================================
;#
;# Model run microdata: entity name and attributes to store at each model run
;#
;# run time list of attributes can include less attributes than entity have
;# for example, full list of Person attributes is:
;
; Person = age, ageGroup, sex, income, salary, salaryGroup, fullTime, isOldAge, pension
;
;# order of attributes is not important, it is defined by entity metadata and cannot be changed at run time
;
[Microdata]

; Person = ageGroup,sex,age,income,isOldAge,pension

; Store all non-internal attributes of Person entity
;
; Person = All

; Store all non-internal attributes of all entities
; NOT recommended for production, use for debug only
;
; All    = true

; Allow to store entities internal attributes
; NOT recommended for production, use for debug only
;
; UseInternal = true

; Write microdata entity attributes into database
; Important: each microdata entity MUST have unique key
;
```

```
; ToDb  = false

; Write microdata entity attributes and events (if enabled) into CSV file(s)
; each microdata entity is written in it's own file
;
; ToCsv = false

; Write microdata entity(s) attributes and events (if enabled) into model Trace output
; Trace output must be enabled to produce any results;
; see Trace options in [OpenM] section above
;
; ToTrace = false

; Write selected events into Trace or CSV file
;
; Events = Birth,Union,Death

; If true then do not write event name into CSV file
;
; NoEventColumn = true
```

Number of sub-values stored in `run_lst.sub_count` table:

```
run_id     model_id    run_name                          sub_count   sub_started  sub_completed  create_dt
----------  ----------  ---------------------------------  ----------  -----------  -------------  ------------------------
11          1           ModelOne 2013-05-30 22:50:29.0447  4           4            4              2013-05-30 22:50:29.0447
```

OpenM++ Compiler (omc) Run Options

Home > Model Development Topics > **OpenM++ compiler arguments and options**

This topic documents the arguments and options of the OpenM++ compiler (omc). These arguments and options are normally used indirectly by build system components shipped with OpenM++ for the supported development environments.

Related topics

- Model Code
- File-based Parameter Values: Representing parameter values in files
- ini File Format

Topic contents

- Overview
- Omc ini-file options

Overview

There are a number of options which control model compilation and publishing. The most frequently used are:

- model name
- input directory containing model .ompp or .mpp source files
- input directory with model parameters (a.k.a. "scenario" .dat files or parameters .csv files)
- input scenario name

The OpenM++ compiler (omc) gets run options in the following order:

- as command line arguments
- from options ini-file
- use default values

Following options are supported by omc command line:

- `-Omc.ModelName` name/of/model/executable, e.g. RiskPaths
- `-Omc.ScenarioName` name/of/base/scenario, e.g. Base, it can be list of names
- `-Omc.InputDir` input/dir/to/find/source/files
- `-Omc.OutputDir` output/dir/to/place/model/cpp_and_h_and_sql/files
- `-Omc.UseDir` use/dir/with/ompp/files
- `-Omc.ParamDir` input/dir/to/find/parameter/files/for/scenario, it can be list of directories
- `-Omc.FixedDir` input/dir/to/find/fixed/parameter/files/
- `-Omc.CodePage` code page for converting source files, e.g. windows-1252
- `-Omc.SqlDir` sql/script/dir to create model SQLite database
- `-Omc.SqliteDir` output directory to create model SQLite database
- `-Omc.SqlPublishTo` create sql scripts to publish in `SQLite,MySQL,PostgreSQL,MSSQL,Oracle,DB2`, default: `SQLite`
- `-OpenM.IniFile` some/optional/omc.ini
- `-Omc.TraceScanning` detailed tracing from scanner
- `-Omc.TraceParsing` detailed tracing from parser

- `-Omc.MessageLanguage` language to display output messages, default: user environment settings
- `-Omc.MessageFnc` localized message functions, default: `LT,logMsg,logFormatted,WriteLogEntry,WarningMsg,ModelExit`
- `-Omc.SqlDir` sql/script/dir to create SQLite database
- `-Omc.SqlPublishTo` create sql scripts to publish in `SQLite,MySQL,PostgreSQL,MSSQL,Oracle,DB2`, default: `SQLite`

Or you can use short form of command line arguments:

- `-m` short form of `-Omc.ModelName`
- `-s` short form of `-Omc.ScenarioName`
- `-i` short form of `-Omc.InputDir`
- `-o` short form of `-Omc.OutputDir`
- `-u` short form of `-Omc.UseDir`
- `-p` short form of `-Omc.ParamDir`
- `-f` short form of `-Omc.FixedDir`
- `-ini` short form of `-OpenM.IniFile`

Each option has a unique key string associated with it, i.e.: `Omc.InputDir`. You can use this key to specify either as a command line argument or in an ini-file Section.Key entry. For example:

```
omc.exe -m RiskPaths -Omc.InputDir ../code -ini my-omc.ini
```

would compile model `RiskPaths` source files: `../code/*.ompp` and `../../code/*.mpp` with some other options specified through my-omc.ini file.

Omc do compile model source *.ompp and* .mpp files and create model.sqlite database with parameter values from *.odat,* .dat, *.csv,* .tsv and *.md files:

```
omc.exe -m RiskPaths -i ../code -s Default -p ../parameters/Default
```

Command above will read *.odat,* .dat, *.csv,* .tsv and *.md files from `../parameters/Default` directory and create `RiskPaths.sqlite` database with `Default` input set of parameters (`Default` scenario).

It is possible to create multiple input sets of parameters (multiple scenarios) when you are building the model:

```
omc.exe -m RiskPaths -i ../code -s Default,Other -p ../parameters/Default,../parameters/other/dir
```

Above command will create two input sets of parameters:

- scenario `Default` from *.dat,* .odat, *.csv,* .tsv and *.md files in `../parameters/Default` directory
- scenario `Other` from *.csv,* .tsv and *.md files in `../parameters/other/dir`

Please note that the second or subsequent scenario directory (if present) can contain only CSV or TSV and Markdown files and not .dat or .odat files.

For more information on specifying parameter values using .csv or .tsv files, please see the topic File-based Parameter Values.

For more information on specifying parameter values using .dat or .odat files, please refer to Modgen documentation.

[back to topic contents]

Omc ini-file options

To specify name of ini-file you can use `-ini` or `-OpenM.IniFile` command line option. Please see OpenM++ ini-file format to find out more.

Example of omc ini-file:

```
;
```

```
; This is an example of omc.ini options file
;

;
; Omc-specific options
;
[Omc]

;
; model name, it must be specified either on command line or here
; no default value
;
; ModelName = NewCaseBased

;
; input directory to get source .ompp or .mpp files to compile
; default = current directory
;
; InputDir = ./code

;
; output directory to place generated .cpp and .h files for the model
; default = current directory
;
; OutputDir = ./src

;
; use directory to resolve 'use' statements
; default = directory/of/omc.exe/../use/
;
; UseDir = ../../use

;
; name of default set of input parameters (a.k.a. base scenario data)
; it can be multiple names separated by comma or semicolon
;
; default = Default
;
; ScenarioName = Default
; ScenarioName = Default,Other,Test

;
; parameter directory to get source .dat or .csv files to publish a scenario
; it can be multiple directories separated by comma or semicolon
;
; default = Default
;
; ParamDir = ../parameters/Default
; ParamDir = ../parameters/Default,../parameters/Other/dir,../parameters/some/Test

;
; fixed directory to get source .dat files with fixed parameter values
; default = Fixed
;
; FixedDir = ../parameters/Fixed

;
; directory where common sql scripts located (used to create SQLite database)
; default = directory/of/omc.exe/../sql/
;
; SqlDir = ../../sql

;
; output directory to create model.sqlite database
; default: value of OutputDir (see above)
;
; SqliteDir = ./src

;
; database providers comma-separated list
; supported providers: SQLite,MySQL,PostgreSQL,MSSQL,Oracle,DB2
; default: SQLite
;
; SqlPublishTo = SQLite

;
; code page for converting source files into utf-8
; default on Linux: utf-8 (no conversion)
; default on Windows: current user code page, e.g.: windows-1252
;
; CodePage = windows-1252

; language to display output messages
; default: Windows Control Panel or Linux LANG
;
; messageLang = en-CA

;
```

```
;
; localized message functions
; first argument of the Function("const char * message",...) translated into other language
; by lookup in omc.message.ini where "message" = "translated message"
; default: LT,logMsg,logFormatted,WriteLogEntry,WarningMsg,ModelExit
;
; MessageFnc = LT,logMsg,logFormatted,WriteLogEntry,WarningMsg,ModelExit

;
; Common openM++ run options supported by omc
;
[OpenM]

;
; log settings:
; log can be enabled/disabled for 3 independent streams:
;   console         - cout stream
;   "last run" file - log file with specified name, truncated on every compiler run
;   "stamped" file  - log file with unique name, created for every compiler run
;
; "stamped" name produced from "last run" name by adding time-stamp and pid-stamp, i.e.:
;   omc.log => omc.2012_08_17_16_04_59_148.1234.log
;

LogToConsole     = true       ; log to console
LogNoMsgTime     = true       ; if true then do not prefix log messages with date-time
; LogToFile        = false    ; log to file
; LogToStampedFile = false    ; log to "stamped" file
; LogUseTimeStamp  = false    ; use time-stamp in log "stamped" file name
; LogUsePidStamp   = false    ; use pid-stamp in log "stamped" file name
; LogFilePath      = omc.log  ; log file path, default = current/dir/omc.log
; LogSql           = false    ; debug only: log sql statements (reserved, but not used by omc)
```

[back to topic contents]

OpenM++ ini-file format

Home > Common Topics > **OpenM++ ini Files**

OpenM++ components can use ini files to specify options. This topic describes how these ini files are structured.

Related topics

- model run ini file options
- omc ini file options

OpenM++ ini-file format

OpenM++ ini-files are similar to other well-known implementations of ini-files. It is a text file consist of `[sections]` of `key = value` pairs and optional comments. For example:

```
[General]
Cases = 12345 ; number of cases

; openM++ specific options
[OpenM]
SparseOutput = true
```

Ini-file can contain following lines:

- [section] line where section is [anything in square brackets]
- Key = Value lines
- empty lines and comment lines

Value can take multiple lines with \ at the end of the line for continuation.

Value can be a string, integer, double or boolean type. Boolean values:

- True value is any of: "yes", "1", "true" or empty value
- False value is any of: "no" "0", "false" Boolean values are not case sensitive, e.g.: "yes" is same as "YeS" and it is a true value Double values must be in "C" locale, which means using dot as decimals separator, i.e.: -123456.78e+9

Comments are optional and can start from either semicolon or hash sign at any position of the line. You can escape comment separator by putting value in single 'apostrophes' or double "quotes".

Example of ini-file format recorgnized by openM++:

```ini
[Test]    ; section is required, global entries are not allowed
          # this is also a comment
          ; next line is empty value without comment
non =
rem =                                         ; comment only and empty value
val = no comments
dsn = "DSN='server'; UID='user'; PWD='secret';"   ; database connection string example
lst = "the # quick" brown 'fox # jumps ; over'    # use "quote" and 'apostrophe' to escape characters and keep spaces
unb = "unbalanced quote                          ; this is not a comment: it is a value started from " quote

trim = Aname,Bname, \     ; multi-line value joined with spaces trimmed
       Cname,DName        ; result is: Aname,Bname,Cname,DName

; multi-line value started with " quote or ' apostrophe
; right spaces before \ is not trimmed, result is:
; Multi line   text with spaces
;
keep = "Multi line   \
       text with spaces"

; multi-line value started with " quote or ' apostrophe
; result is the same as above:
; Multi line   text with spaces
;
same = "\
       Multi line   \
       text with spaces\
       "

; General settings
[General]
StartingSeed=16807
Subsamples=8
Cases = 5000        ; only for case-based
SimulationEnd = 100 ; only for time-based
UseSparse = true

#
# override values of above [Test] section with new values
#
[Test]
val=new value of no comments
dsn="new value of UID='user'; PWD='secret';"       ; new database connection string
lst=new value of "the # quick" fox 'jumps # over'  # new list of test words
```

UI: How to start user interface

How to use openM++ UI

Open++ user interface (ompp-ui) is a lightweight web UI which is:

- scalable: can be run on single end-user desktop and in cluster environment
- cloud ready: can be deployed in private or public cloud (Amazon AWS, Microsoft Azure, Google Cloud, etc.)
- portable: work on Windows, Linux and MacOS, 32 and 64 bit versions
- open source: it is open source product

Start openM++ UI

By default ompp-ui does not require any installation, to run it do one of the following:

- on Windows double click on `bin\ompp_ui.bat`
- on Linux double click on `bin/ompp_ui.sh`
- on MacOS double click on `bin/ompp_ui.command`

Any of above script is relatively simple, all it does is starting oms web-service:

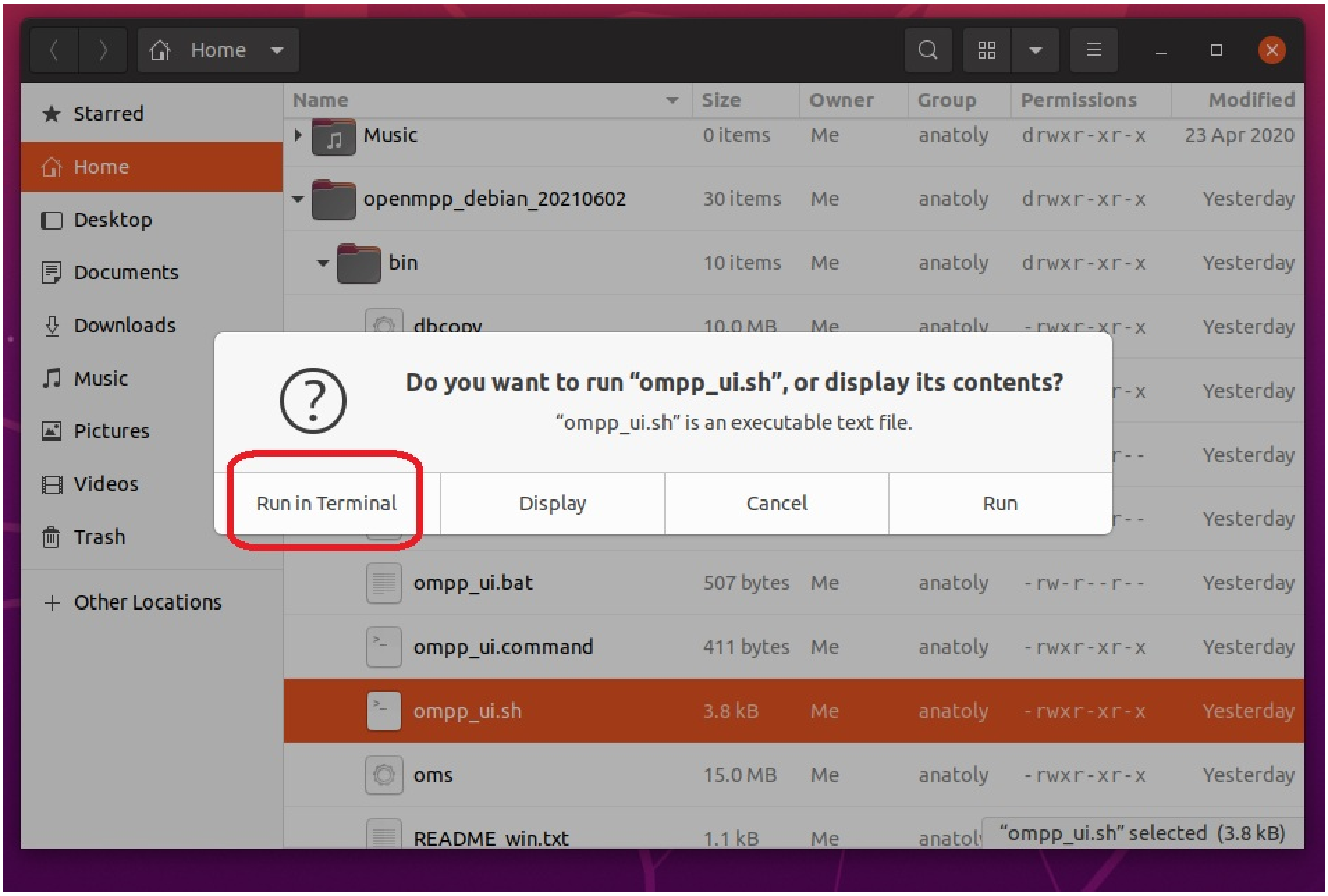
```
cd ~/openmpp_mac_20200704
bin/oms
.....
2020-06-19 16:07:57.892 Model directory: models/bin
2020-06-19 16:07:57.930 Listen at localhost:4040
2020-06-19 16:07:57.930 To start open in your browser: localhost:4040
2020-06-19 16:07:57.931 To finish press Ctrl+C
```

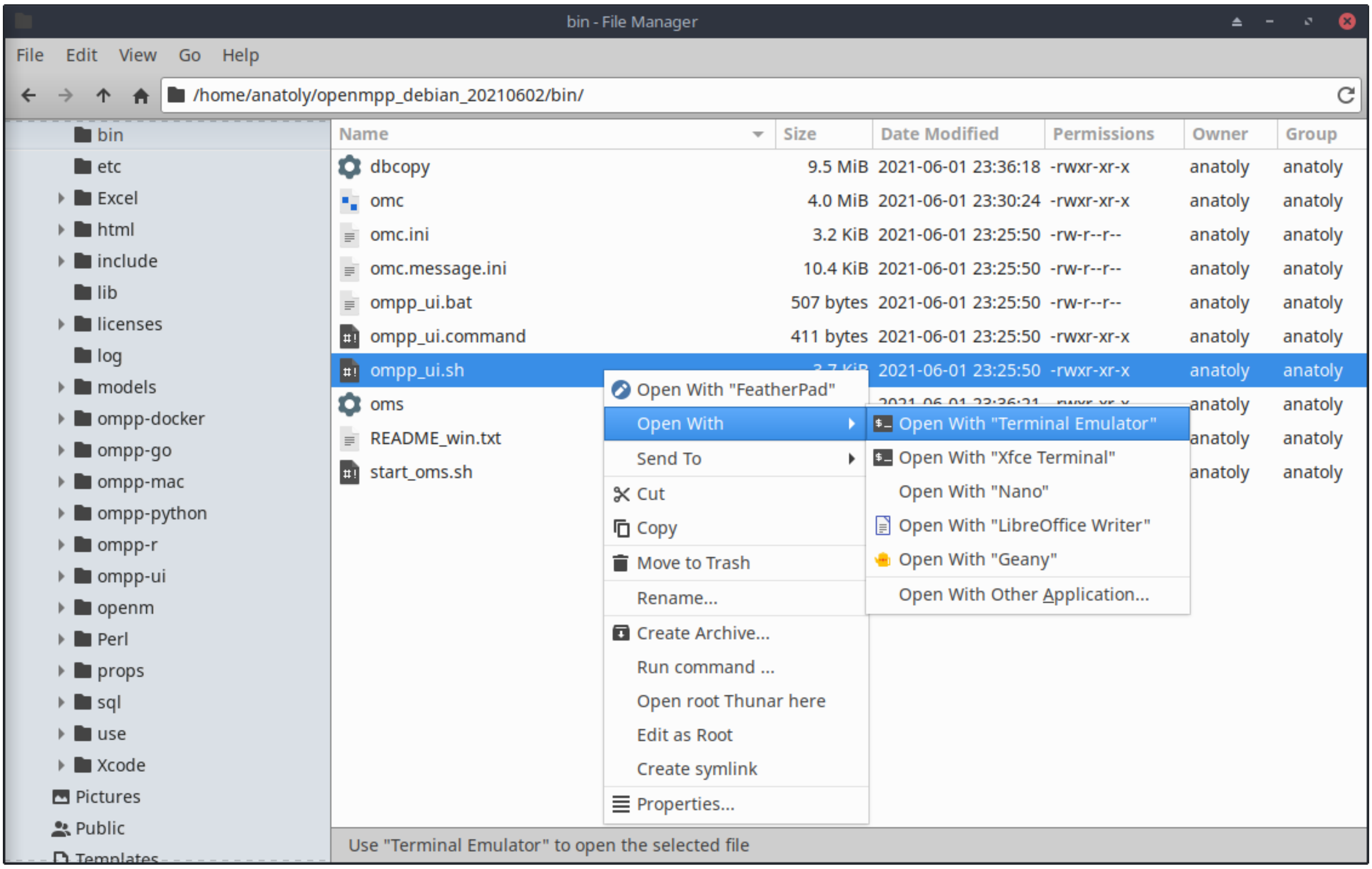
and open your browser at `http://localhost:4040`

Linux: Not every distribution do run executable by double click, if this action does not work then do it from command line:

```
cd openmpp_debian_20200704
./bin/ompp_ui.sh
```

It is possible you will be asked to confirm or select the action "Run in terminal" or "Open with Terminal":





Use model runs queue

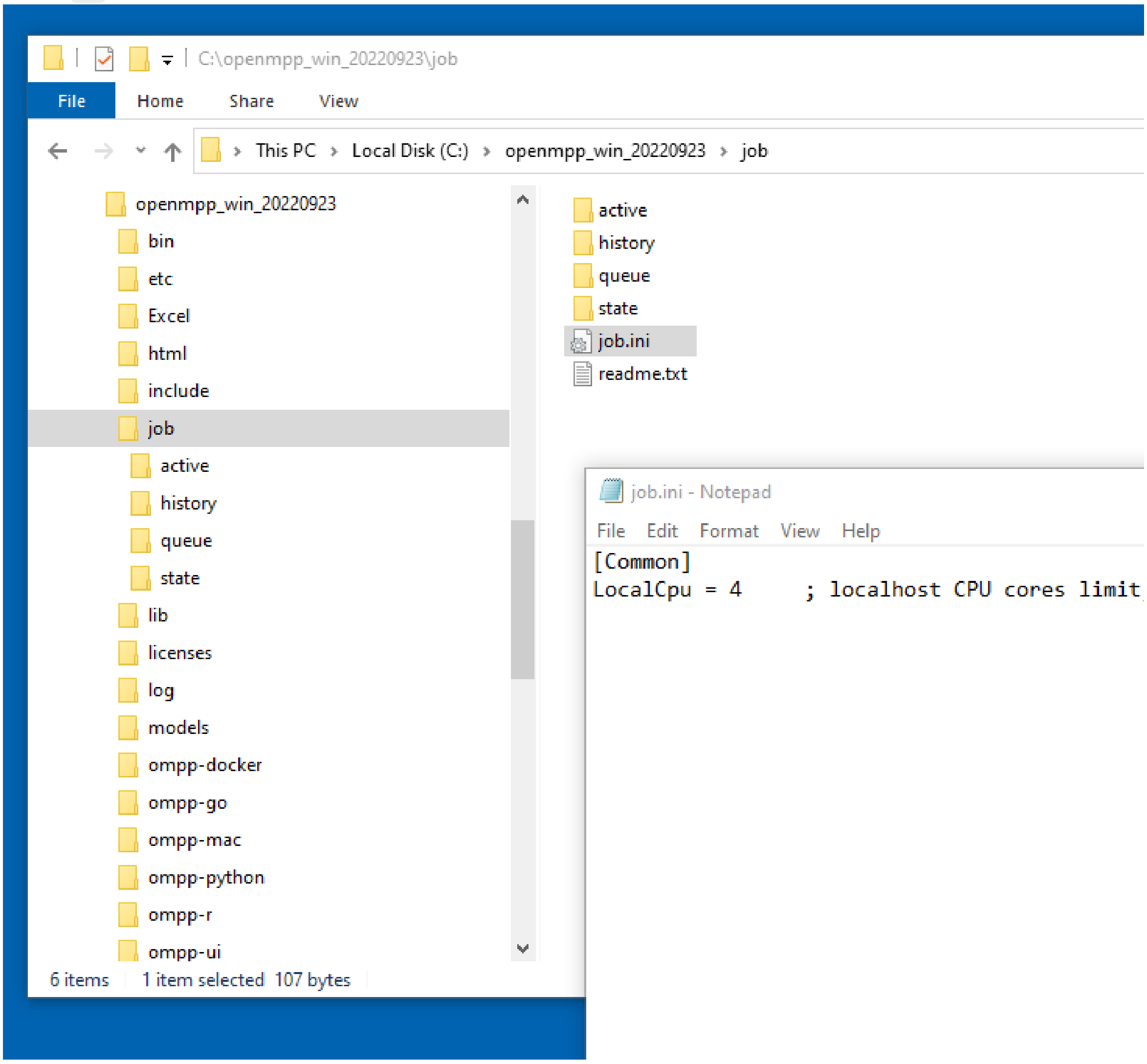
If model runs queue enabled then from UI Service Status page you can:

- see current model runs queue

- re-arrange your model run queue
- delete your model run job from the queue
- see the history of model runs
- re-submit model run again if it was failed
- see currently active model runs
- cancel (stop) model run

In order to enable model runs queue on your local computer do following:

- create `job` directory and sub-directories under your openM++ installation folder:



You can just copy `job` directory, sub-directories and `job.ini` from `ompp-go` folder of your openM++ installation.

- edit `job.ini` file to specify number of CPU cores which you want to use for model run, e.g.:

```
[Common]
LocalCpu = 4
```

- modify UI start script to add `-oms.JobDir job` option to the `oms` line
  - on Windows `bin\ompp_ui.bat`:

    ```
    ...skip... \bin\oms -oms.HomeDir models\home -oms.AllowDownload -oms.AllowUpload -oms.LogRequest -oms.JobDir job
    ```

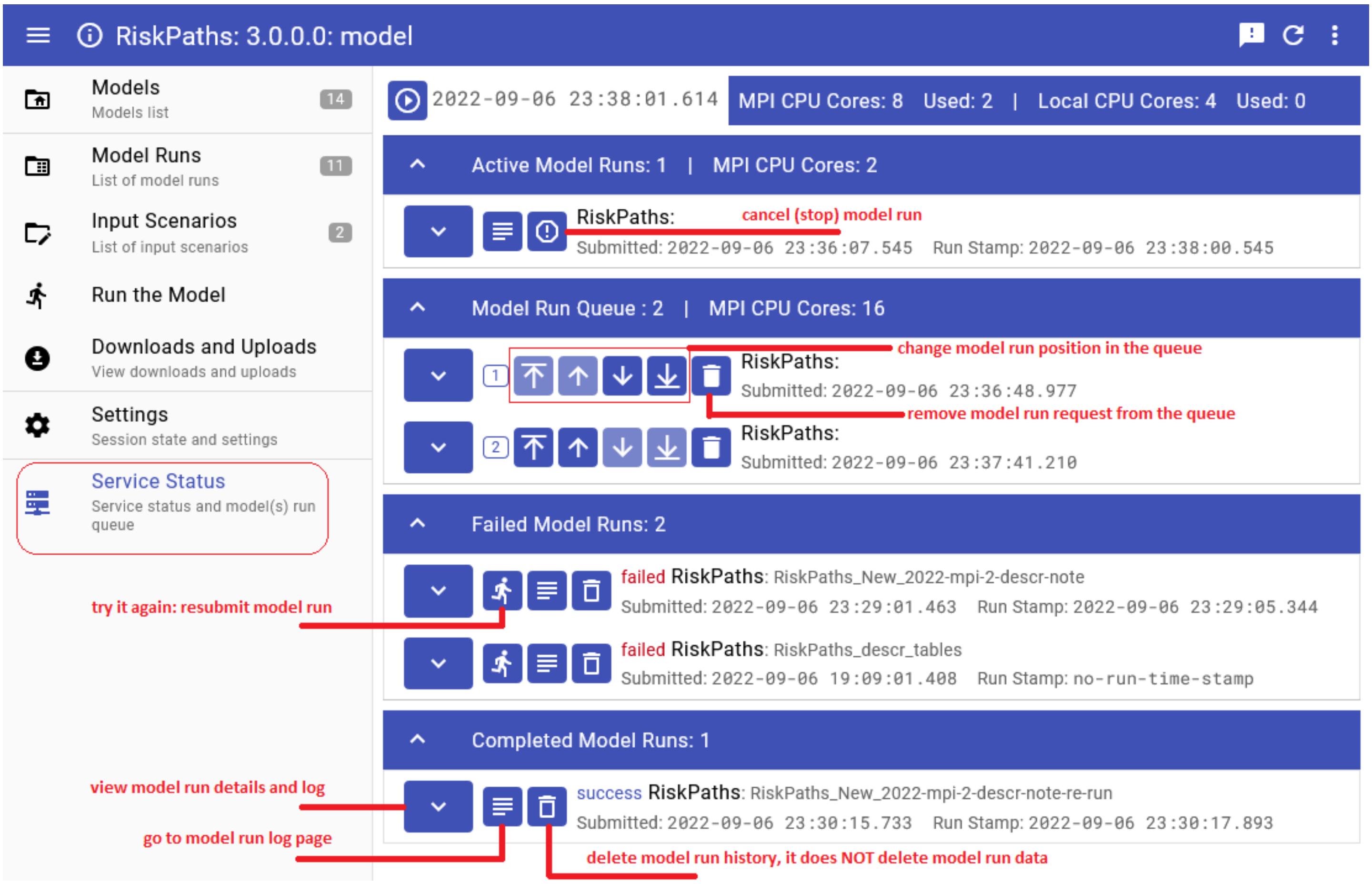
  - on Linux `bin/start_oms.sh`:

```
...skip... ./bin/oms -l localhost:${OMS_PORT} -oms.HomeDir models/home -oms.AllowDownload -oms.AllowUpload -oms.LogRequest -oms.JobDir job
```

- on MacOS `bin/ompp_ui.command`:

```
"...skip... bin/oms -l localhost:4040 -oms.HomeDir models/home -oms.AllowDownload -oms.AllowUpload -oms.LogRequest -oms.JobDir job"
```

After that you can start UI by double click on `bin\ompp_ui.bat` (Windows) or `bin/ompp_ui.sh` (Linux) or `bin/ompp_ui.command` (MacOS). Model runs queue and status page will look similar to:



Start openM++ UI from model source directory

Linux: To start UI from your model source code directory:

```
cd openmpp_debian_20211130/models/RiskPaths
../start-ompp-ui-linux.sh
```

If you `make RELEASE` model then it may be convenient to use one of the following:

```
RELEASE=1 ../start-ompp-ui-linux.sh
export RELEASE=1 make all publish && ../start-ompp-ui-linux.sh
```

It is recommended to stop oms web-service after you are done with UI:

```
cd openmpp_debian_20211130/models/RiskPaths
../stop-ompp-ui-linux.sh
```

If your model source code directory located outside of openM++ release directory then do `export OM_ROOT`:

```
export OM_ROOT=$HOME/openmpp_debian_20211130
cd ~/my-models/RiskPaths
$OM_ROOT/models/start-ompp-ui-linux.sh
```

MacOS: To start UI from your model source code directory:

```
cd openmpp_mac_arm64_20211130/models/RiskPaths
../start-ompp-ui-mac.sh
```

If you `make RELEASE` model then it may be convenient to use one of the following:

```
RELEASE=1 ../start-ompp-ui-mac.sh
export RELEASE=1 make all publish && ../start-ompp-ui-mac.sh
```

It is recommended to stop oms web-service after you are done with UI:

```
cd openmpp_mac_arm64_20211130/models/RiskPaths
../stop-ompp-ui-mac.sh
```

If your model source code directory located outside of openM++ release directory then do `export OM_ROOT`:

```
export OM_ROOT=$HOME/openmpp_mac_arm64_20211130
cd ~/my-models/RiskPaths
$OM_ROOT/models/start-ompp-ui-mac.sh
```

Windows: To start UI from your model source code directory:

1. Copy `start-ompp-ui.bat` into your model folder, for example: `C:\openmpp_win_20220105\props\start-ompp-ui.bat => C:\openmpp_win_20220105\models\RiskPaths`
2. Double click on `start-ompp-ui.bat` or from command line window do:

   ```
   cd \openmpp_win_20220105\models\RiskPaths
   start-ompp-ui.bat
   ```

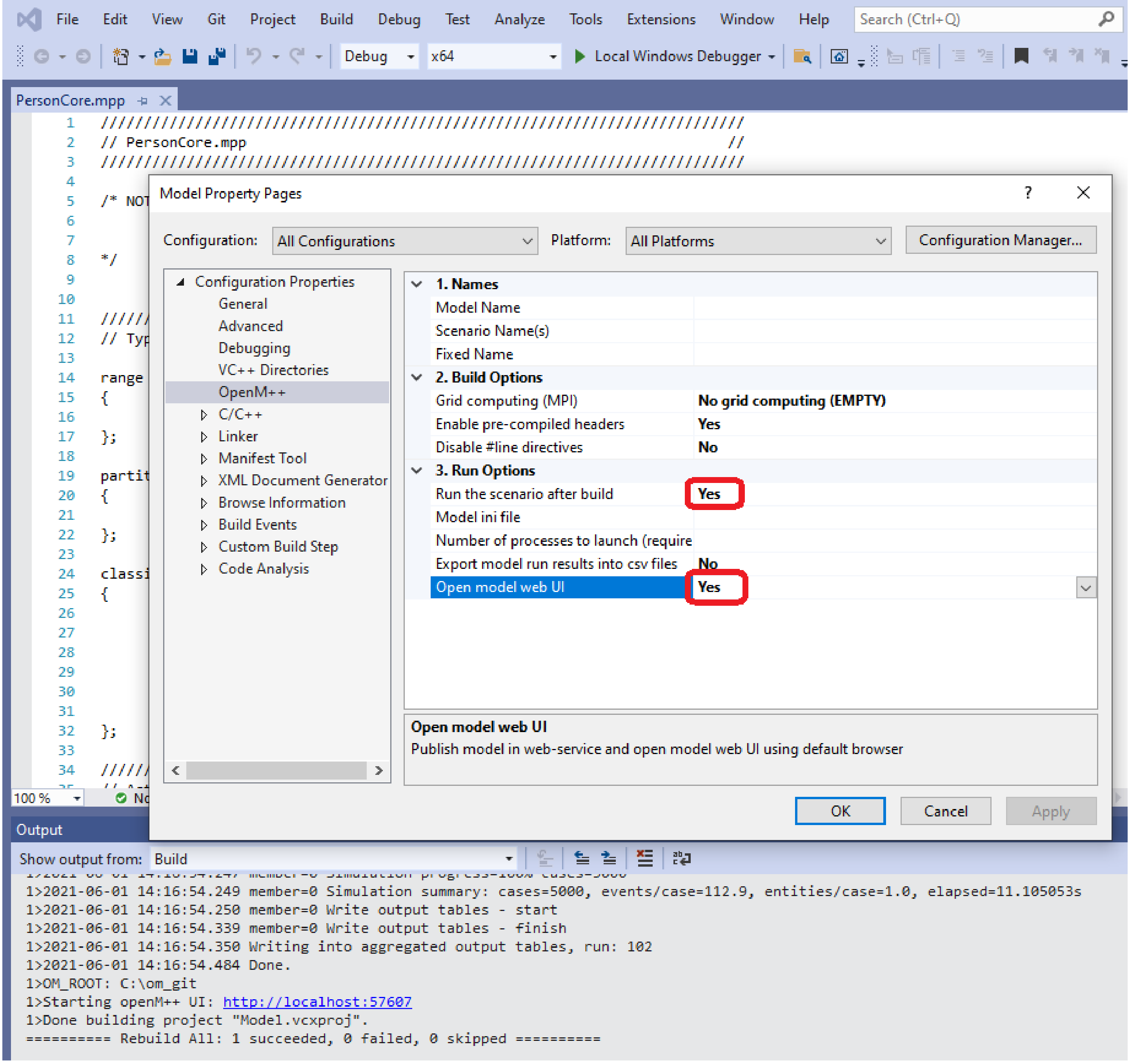
3. It is recommended to close `oms` web-service window after you are done with UI.

If your model source code directory located outside of openM++ release directory then `set OM_ROOT`:

```
set OM_ROOT=C:\openmpp_win_20220105
cd \my-models\RiskPaths
start-ompp-ui.bat
```

Start model UI on Windows from Visual Studio

To open UI from Visual Studio solution model build change project settings as on screenshot below. Optionally you may also want to run the model during model build to see results in UI.



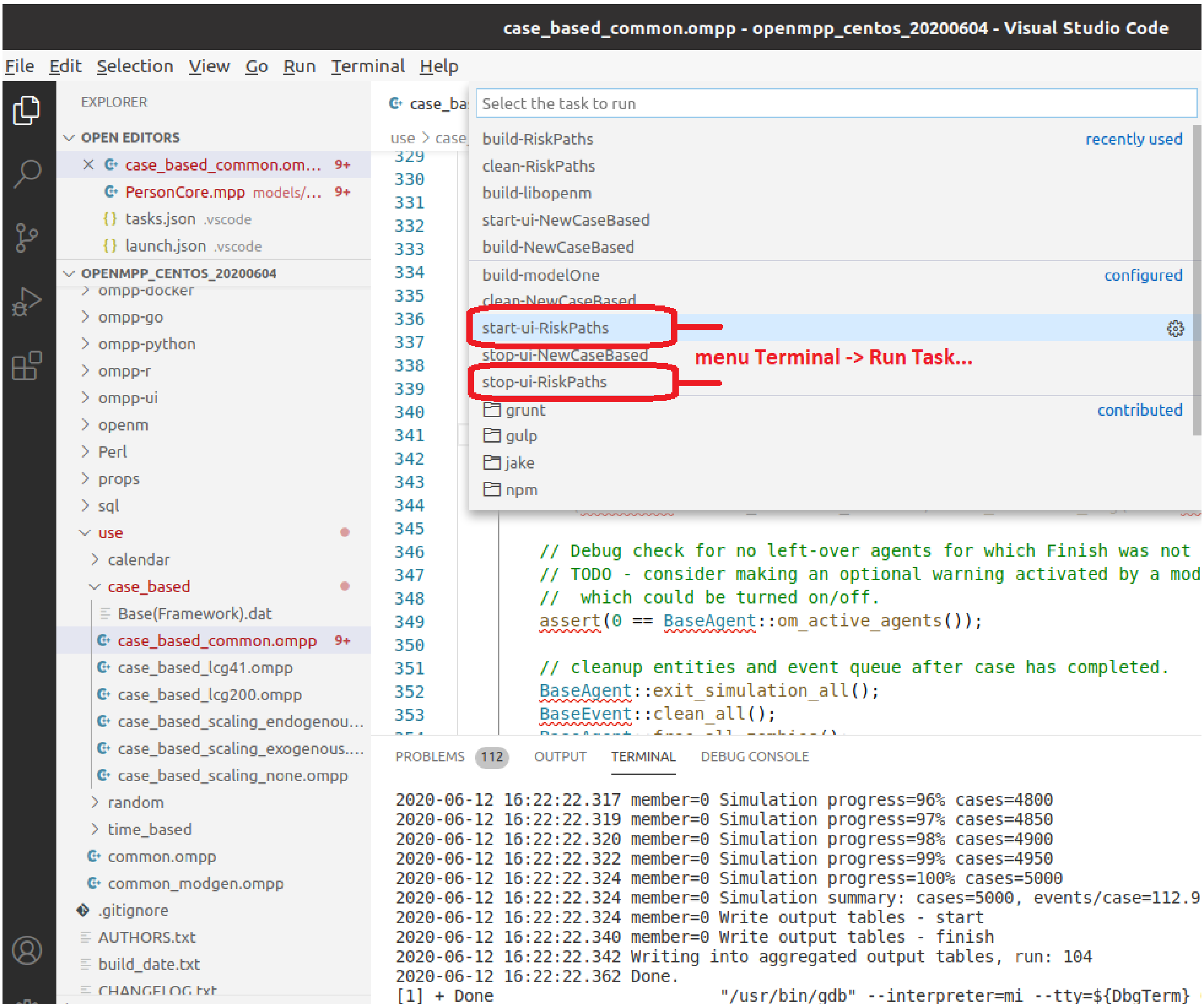
More details about using Visual Studio for model development available at Windows: Create and Debug Models

Start model UI on Linux from Visual Studio Code

To open UI from Visual Studio Code on Linux please configure "Start UI" task for the model. It can be done by using menu Terminal -> Configure Tasks... and create tasks similar to `RiskPaths` model below:

```
{
    // See https://go.microsoft.com/fwlink/?LinkId=733558
    // for the documentation about the tasks.json format
    "version": "2.0.0",
    "tasks": [
        {
            "label": "build-RiskPaths",
            "type": "shell",
            "command": "make all publish",
            "problemMatcher": "$gcc",
            "group": {
                "kind": "build",
                "isDefault": true
            },
            "dependsOrder": "sequence",
            "dependsOn": [
                "build-libopenm",
                "stop-ui-RiskPaths"
            ]
        },
        {
            "label": "start-ui-RiskPaths",
            "type": "shell",
            "command": "../start-ompp-ui-linux.sh",
            "problemMatcher": []
        },
        {
            "label": "stop-ui-RiskPaths",
            "type": "shell",
            "command": "../stop-ompp-ui-linux.sh",
            "problemMatcher": []
        },
        {
            "label": "clean-RiskPaths",
            "type": "shell",
            "command": "make clean-all",
            "group": "build",
            "problemMatcher": []
        },
        {
            "label": "build-libopenm",
            "type": "shell",
            "command": "make libopenm",
            "options": {
                "cwd": "../../openm"
            },
            "problemMatcher": "$gcc",
            "group": "build"
        }
    ]
}
```

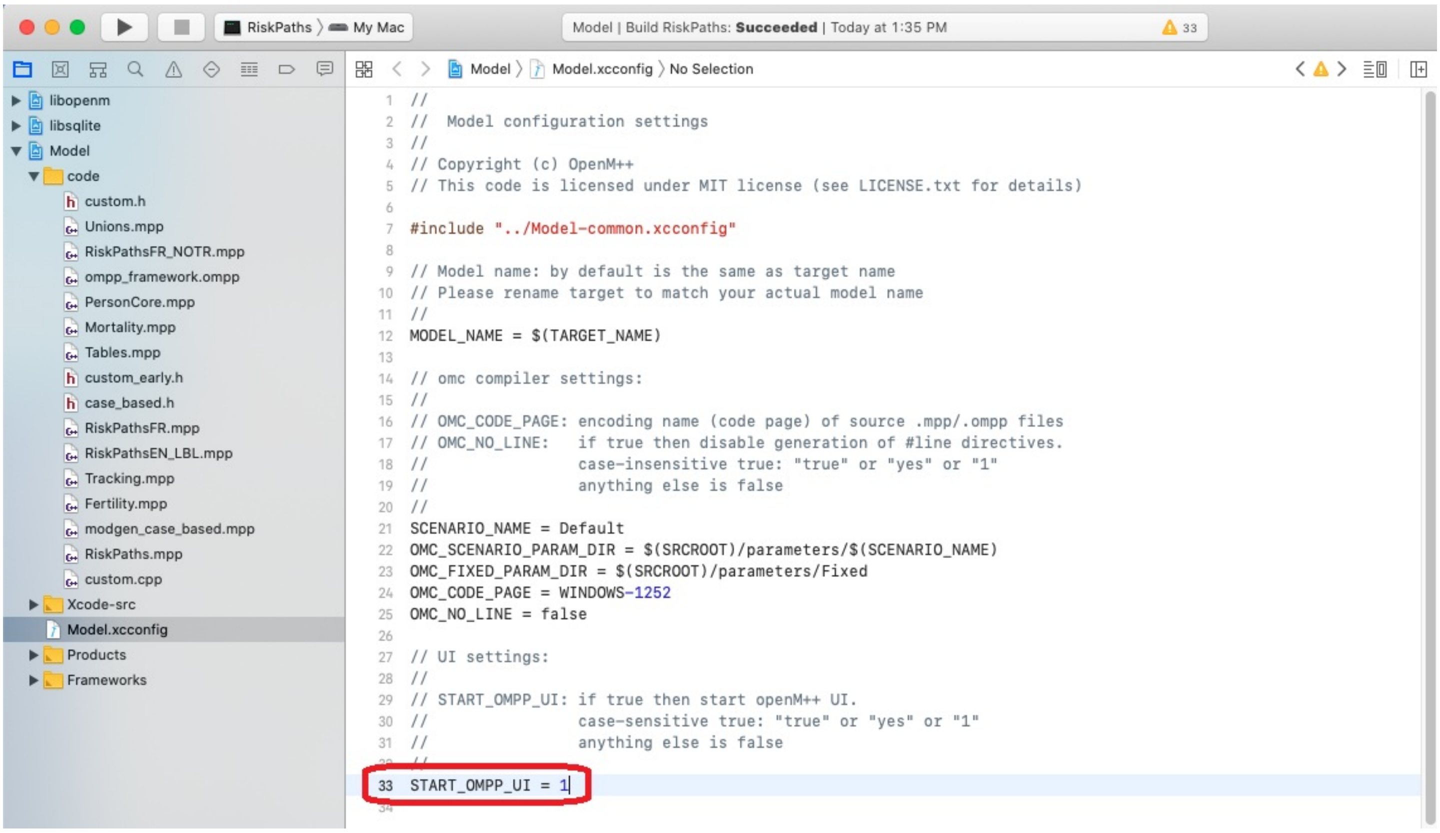
To start UI please go to menu Terminal -> Run Task... -> `start-ui-RiskPaths` After you done with UI it is recommended to shutdown background oms web-service by using Terminal -> Run Task... -> `stop-ui-RiskPaths`



More details about model development on Linux available at Linux: Create and Debug Models

Start model UI on MacOS from Xcode

To start model UI after build completed please change `Model.xcconfig` variable `START_OMPP_UI` to "1" or "true" or "yes" (case-sensitive)



More details about model development on MacOS available at MacOS: Create and Debug Models More details about using Xcode for model development available at MacOS: Create and Debug Model using Xcode

UI: openM++ user interface

Home > **OpenM++ User Interface**

This topic shows functionality of the OpenM++ UI through annotated screenshots. The UI can also be explored by hovering over elements to display short descriptions.

Related topics

- Starting the UI How to start the UI
- Create new scenario or edit existing scenario
- Upload input scenario or parameters
- Run the Model

Topic contents

- Introduction and Background
- Terminology and Concepts
- Screenshot: Chart
- Screenshot: Heat map
- Screenshot: Model runs
- Screenshot: create new scenario or edit existing scenario
- Screenshot: Create new scenario
- Screenshot: Select existing scenario to edit
- Screenshot: Edit parameter
- Screenshot: Run the model
- Screenshot: Compare model runs
- Screenshot: Compare run parameters
- Screenshot: Download model data
- Screenshot: Upload scenario
- Screenshot: Download parameter
- Screenshot: Upload parameter
- Screenshot: Session state and settings

Introduction and Background

The Open++ user interface is a lightweight web UI which can be run from any browser. It is

- scalable: can be run on single end-user desktop and in cluster environment
- cloud ready: can be deployed in private or public cloud (Amazon AWS, Microsoft Azure, Google Cloud, etc.)
- portable: works on Windows, Linux and MacOS, 32 and 64 bit versions
- open source

The OpenM++ UI is an advanced beta which includes significant portions of core functionality but omits others. The underlying software architecture is modern and layered, to make it easy to change or evolve the UI.

Your feedback on the openM++ UI is welcomed. Please feel free to join and participate in discussion of the openM++ UI on GitHub.

[back to topic contents]

Terminology and Concepts

Some key terms:

Term	Meaning
Scenario	A set of one or more parameters and the values of those parameters.
Partial scenario	A scenario which does not include all parameters.
Complete scenario	A scenario consisting of *all* parameters.
Run specification	A completely specified set of parameters. It can be either A) a complete scenario or B) a partial scenario combined with a base run or a complete scenario.
Completed run	All input parameters together with output tables resulting from a model execution.

When a model is first built and published, it includes a complete scenario which is normally named Default. It does not necessarily include a run.

A scenario is best thought of as a *subset* of parameters and their values. Those values are typically modified with respect to some other scenario or run.

A partial scenario cannot be run. It must first be paired with a base run or a complete scenario to supply values for parameters which are absent from the partial scenario. That pairing results in a run specification.

A scenario does not become a run when a run specification uses it or when a run is submitted. Scenarios are independent of runs. For example, the same scenario could be combined with two different base runs to produce two new runs, each with its own run name.

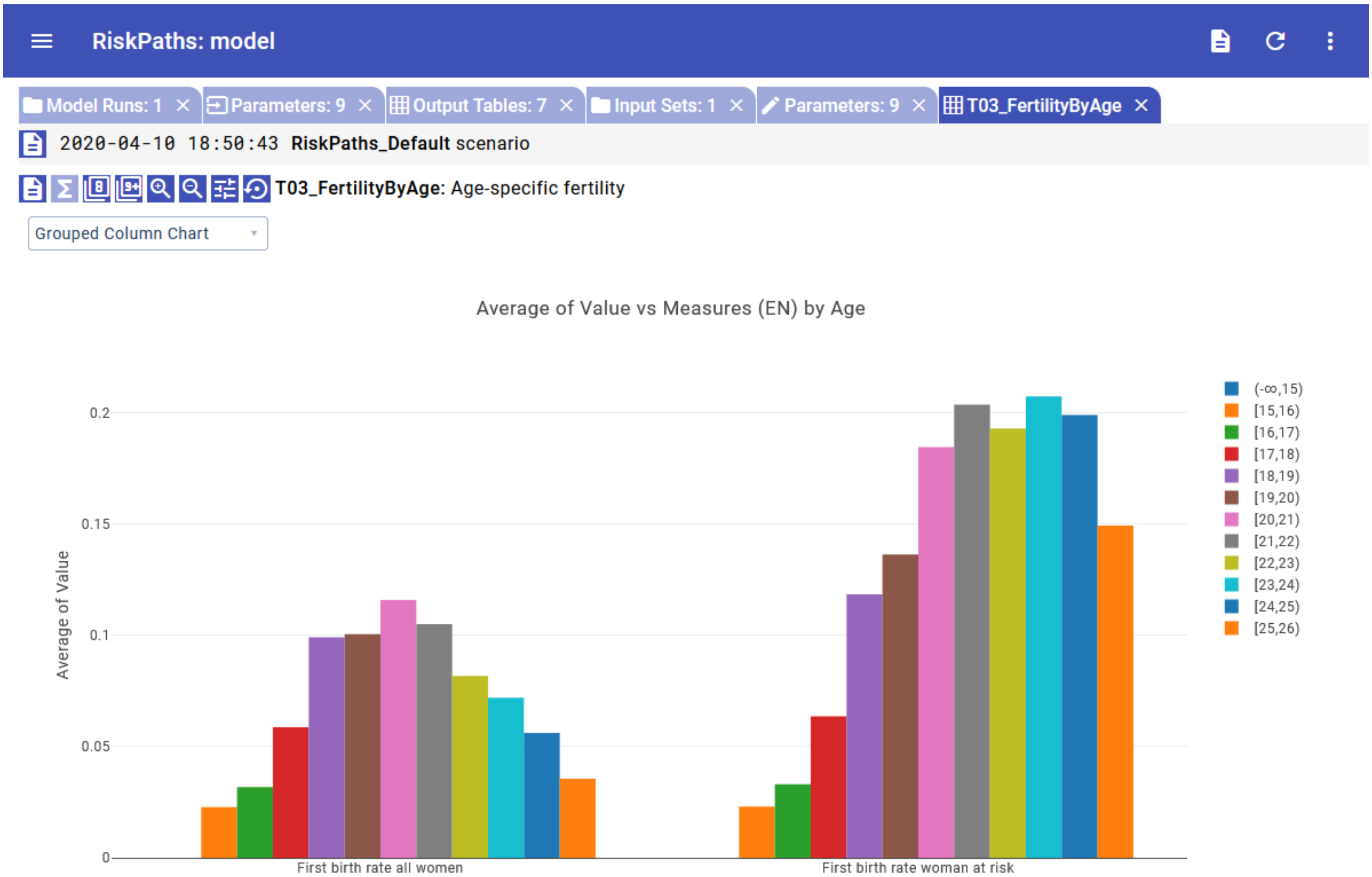
A scenario has a name given when it was created.

A run has a name given when it was specified.

Depending on the names a user chooses, a scenario might have the same name as a run, but it is nevertheless a different kind of object.

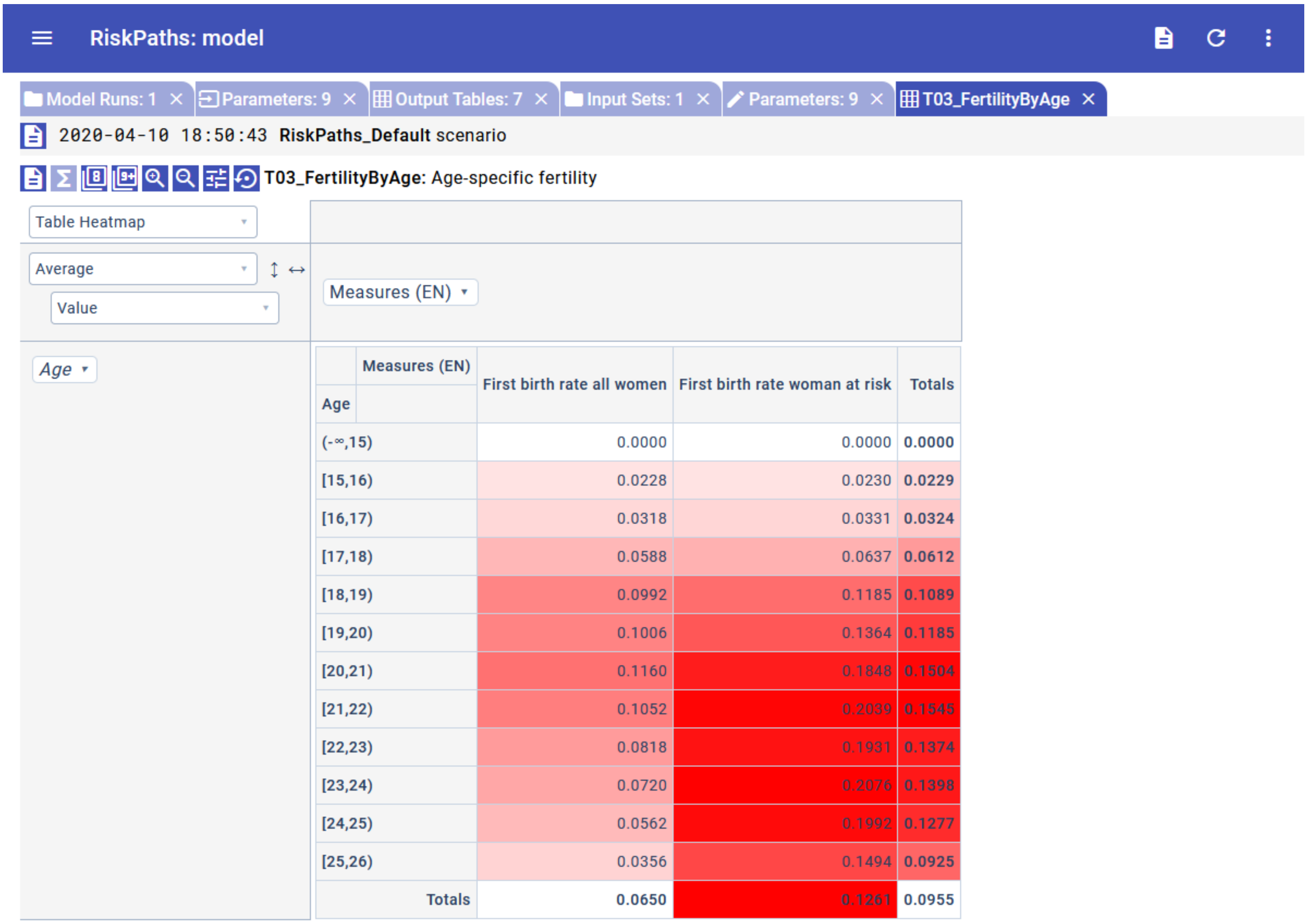
[back to topic contents]

Screenshot: Chart



[back to topic contents]

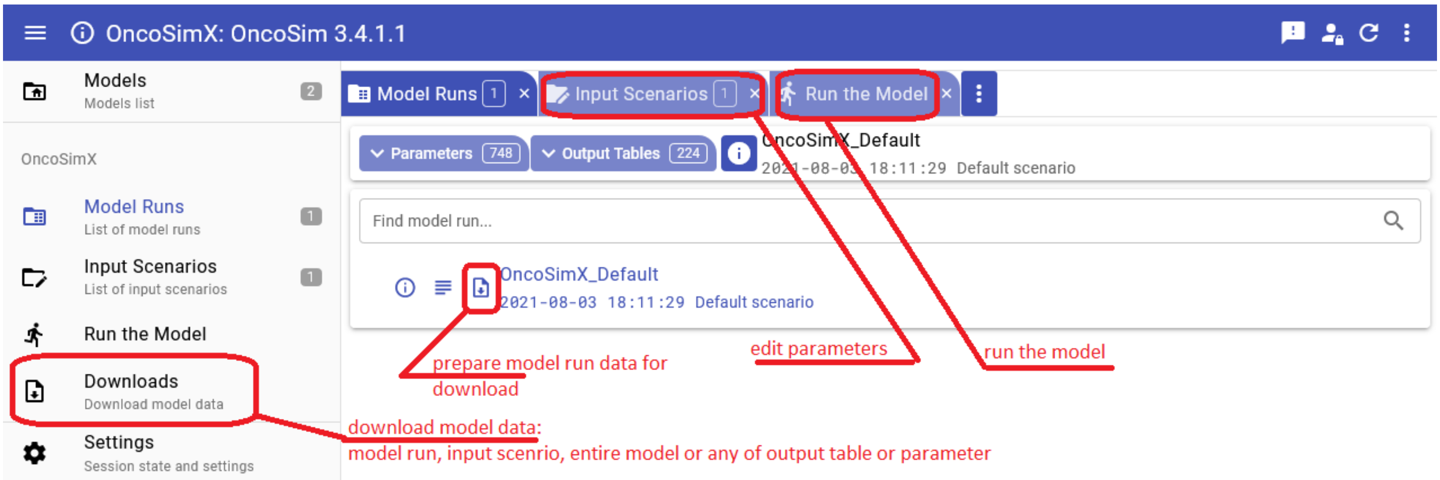
Screenshot: Heat map



Age	First birth rate all women	First birth rate woman at risk	Totals
(-∞,15)	0.0000	0.0000	0.0000
[15,16)	0.0228	0.0230	0.0229
[16,17)	0.0318	0.0331	0.0324
[17,18)	0.0588	0.0637	0.0612
[18,19)	0.0992	0.1185	0.1089
[19,20)	0.1006	0.1364	0.1185
[20,21)	0.1160	0.1848	0.1504
[21,22)	0.1052	0.2039	0.1545
[22,23)	0.0818	0.1931	0.1374
[23,24)	0.0720	0.2076	0.1398
[24,25)	0.0562	0.1992	0.1277
[25,26)	0.0356	0.1494	0.0925
Totals	0.0650	0.1261	0.0955

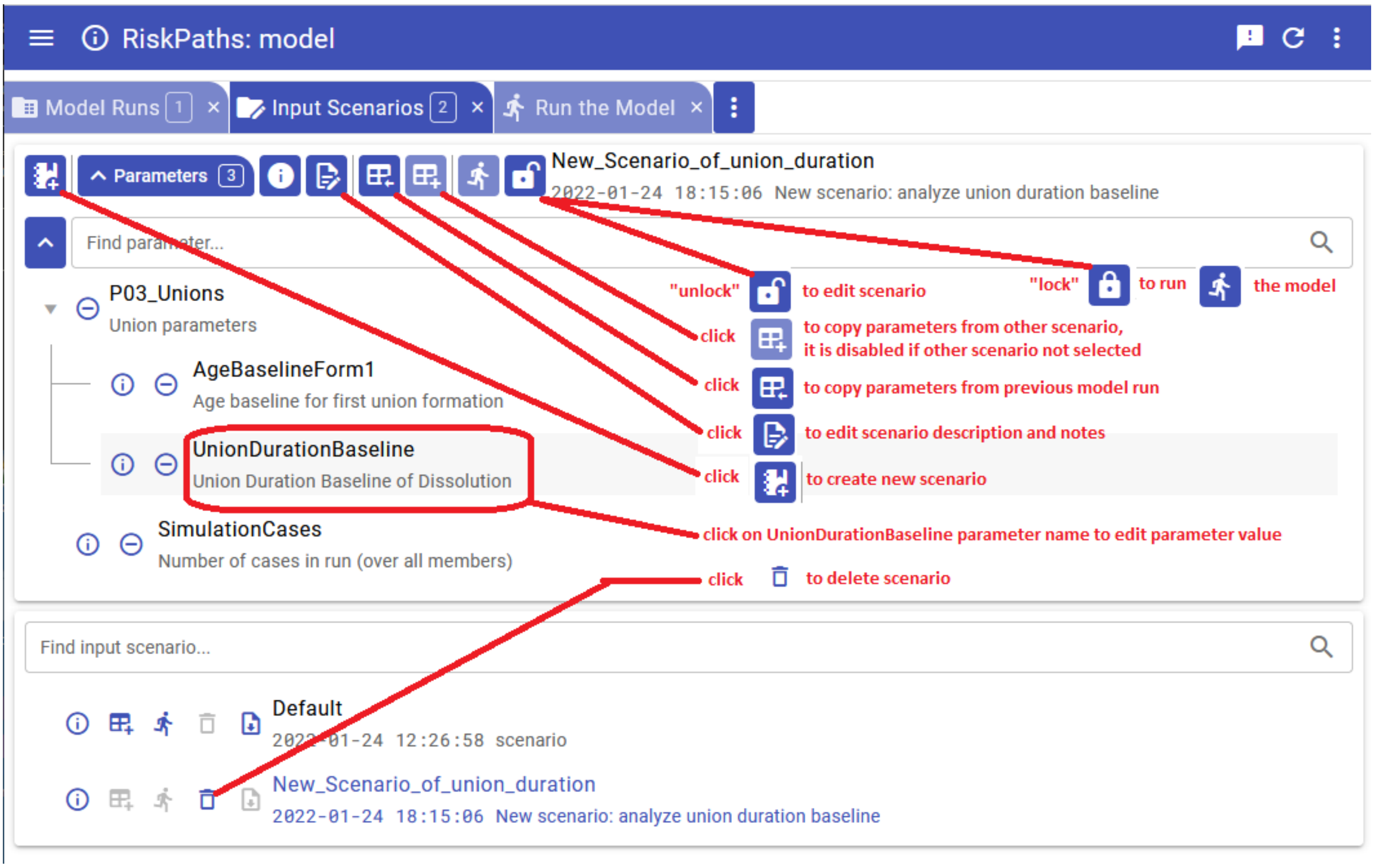
[back to topic contents]

Screenshot: Model runs



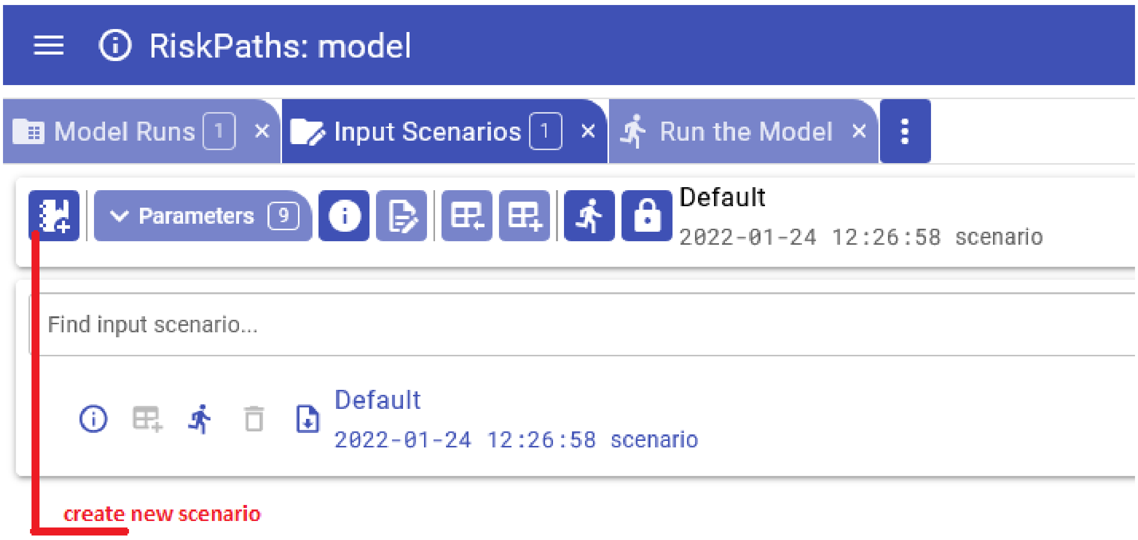
[back to topic contents]

Screenshot: Create new scenario or edit existing scenario



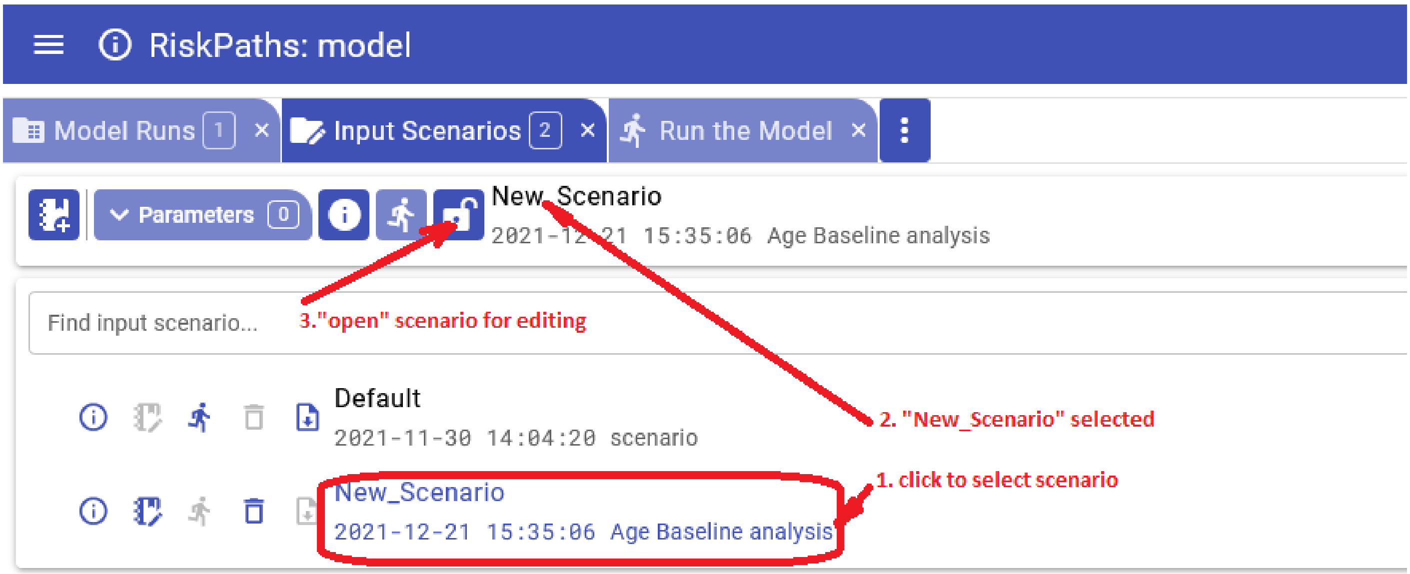
[back to topic contents]

Screenshot: Create new scenario



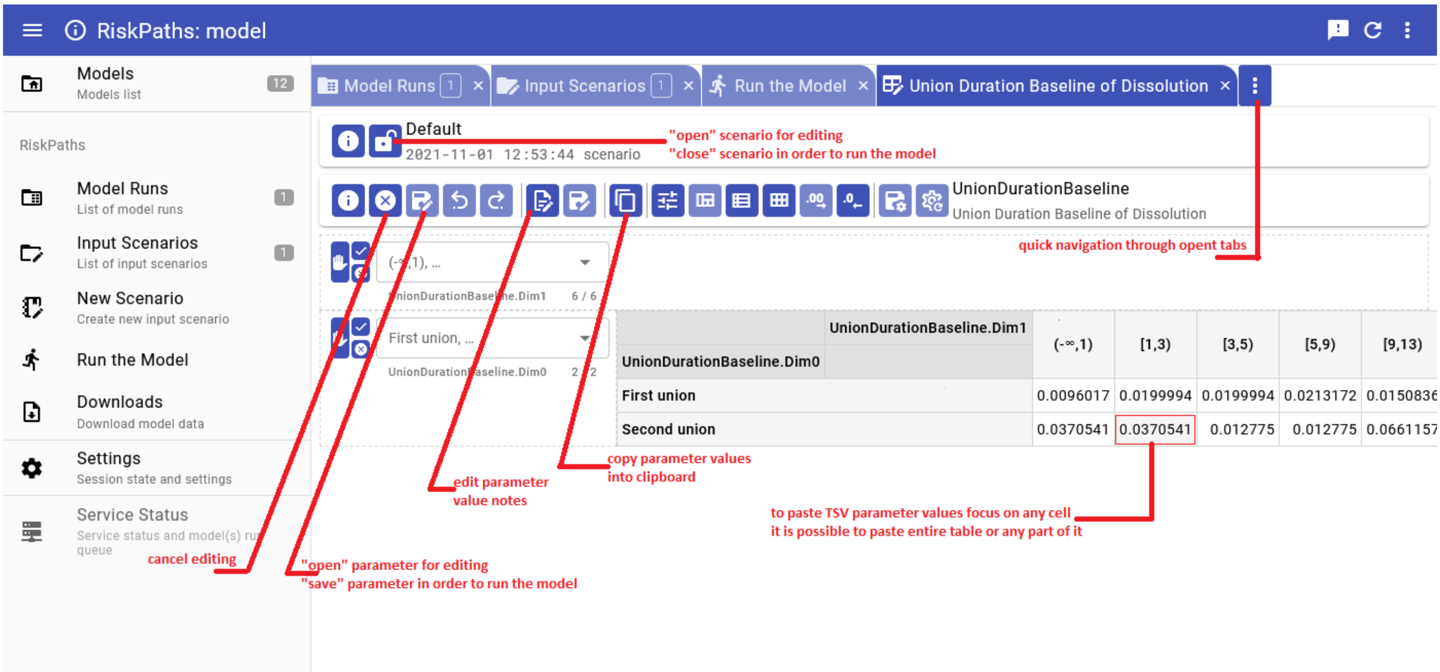
[back to topic contents]

Screenshot: Select existing scenario to edit



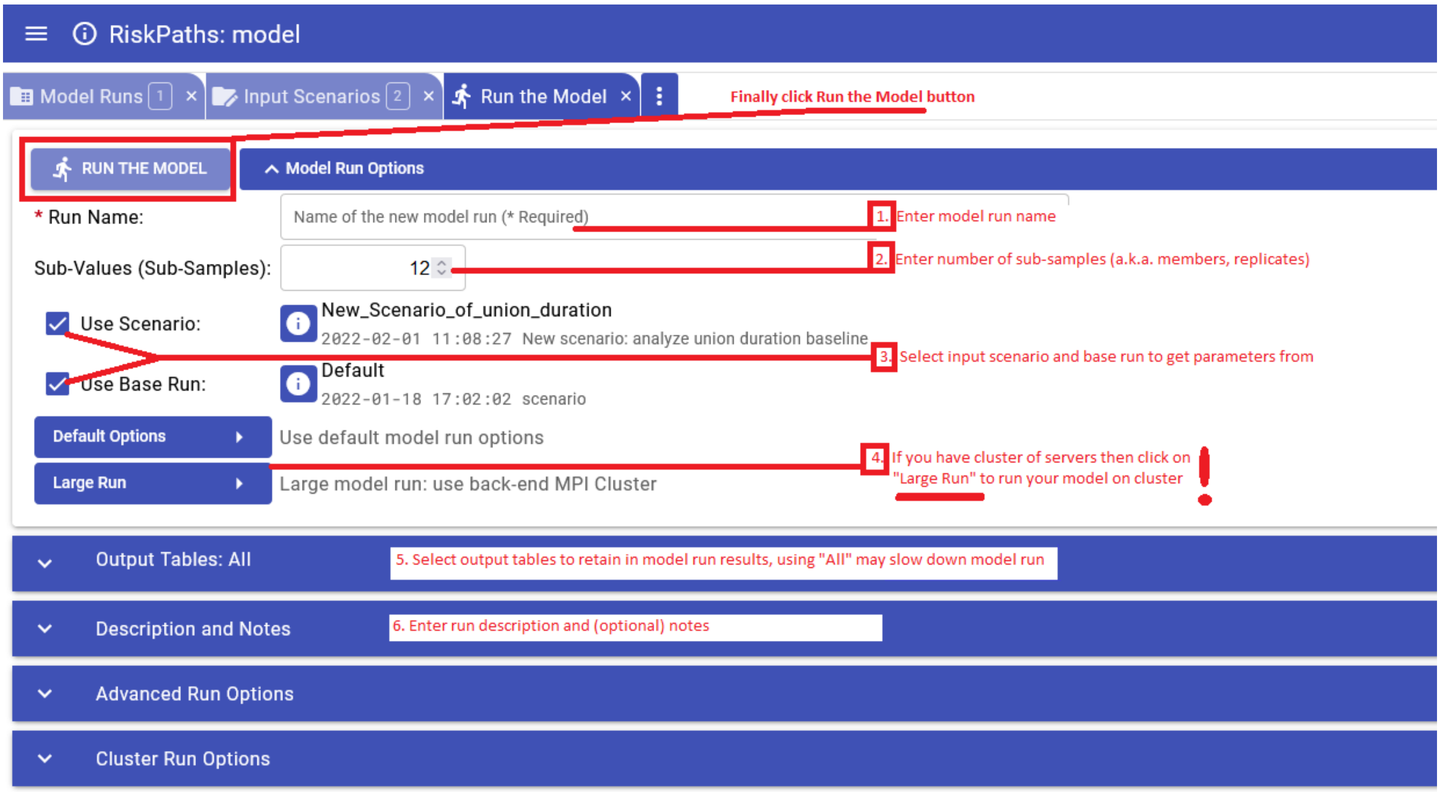
[back to topic contents]

Screenshot: Edit parameter



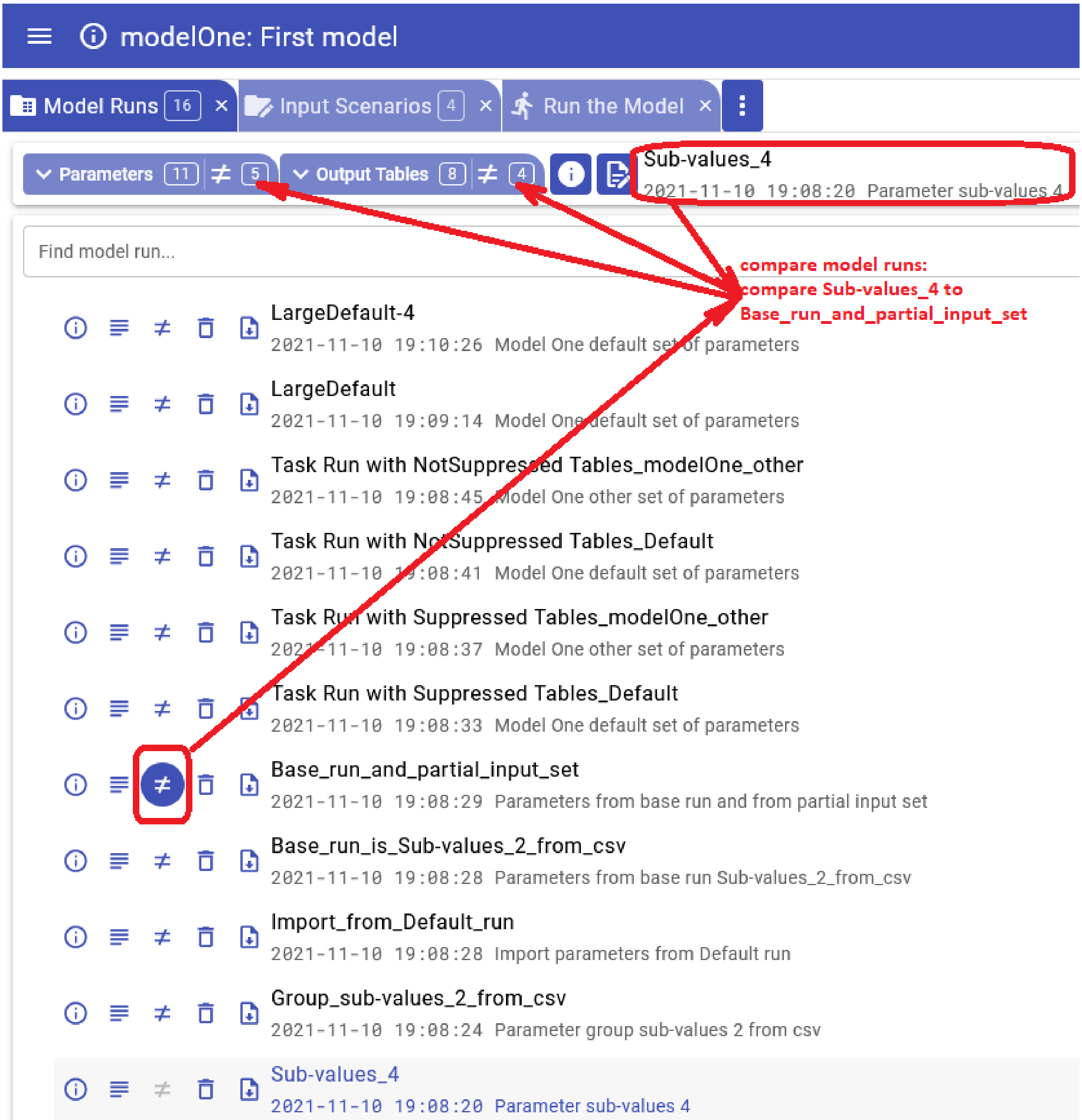
[back to topic contents]

Screenshot: Run the model



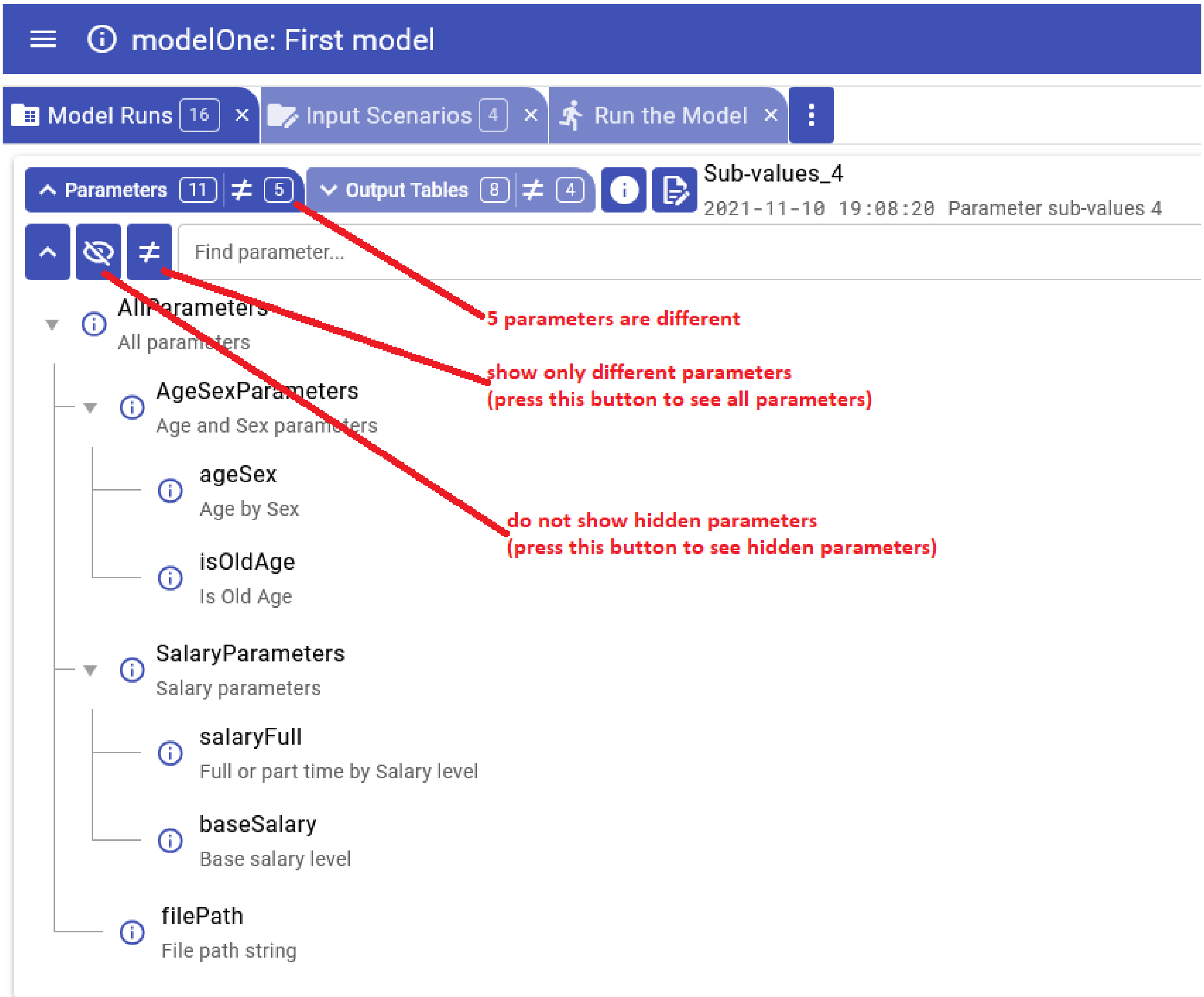
[back to topic contents]

Screenshot: Compare model runs



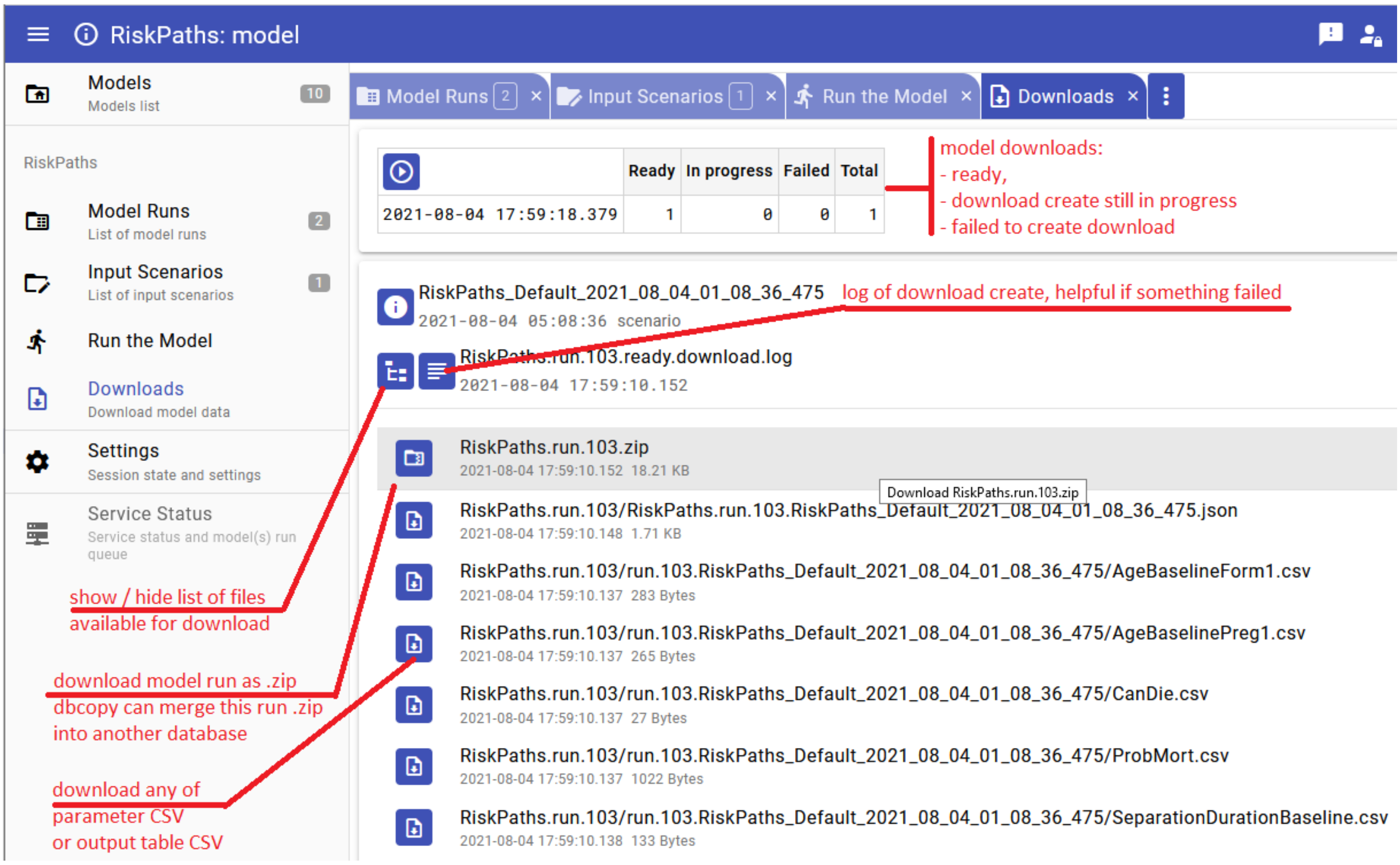
[back to topic contents]

Screenshot: Compare run parameters



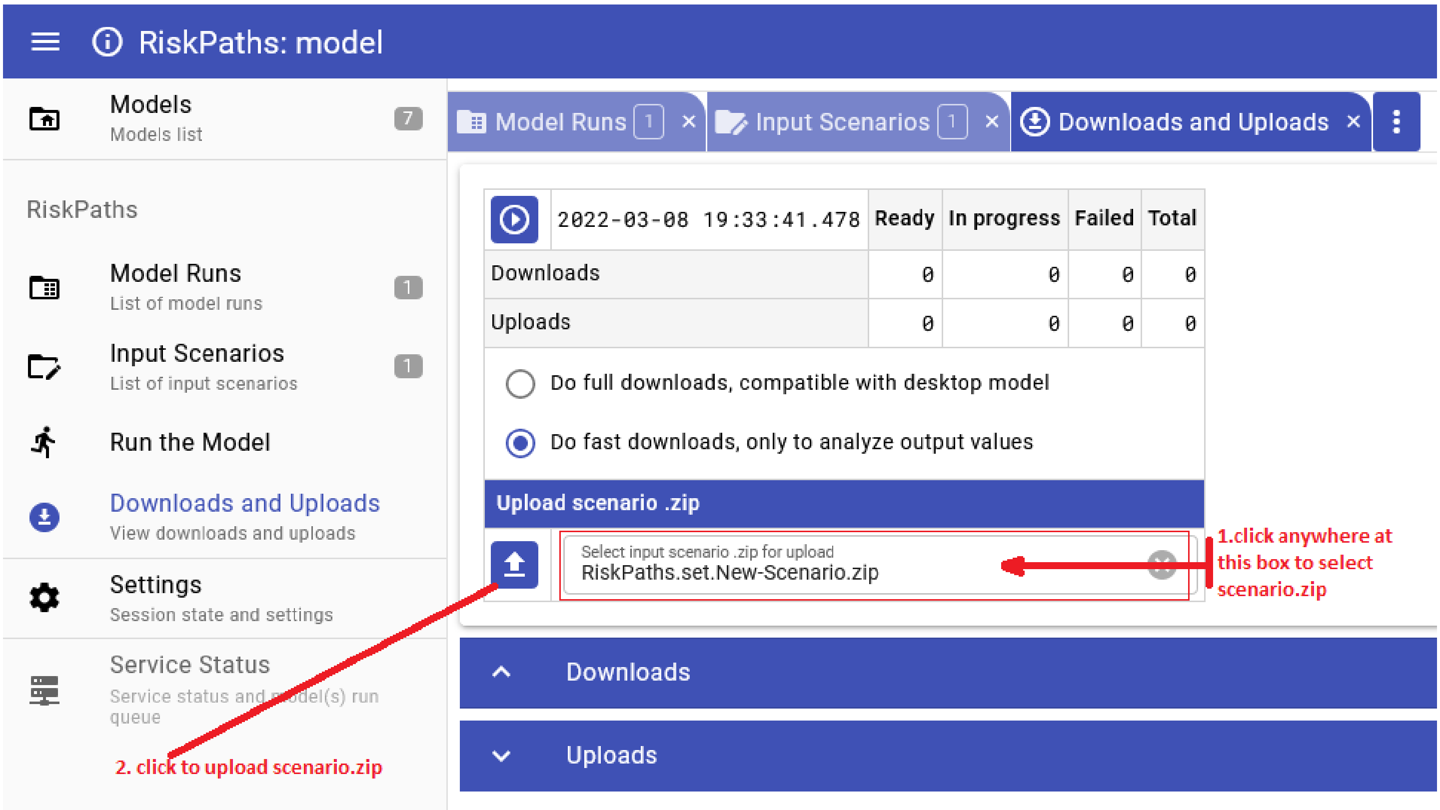
[back to topic contents]

Screenshot: Download model data



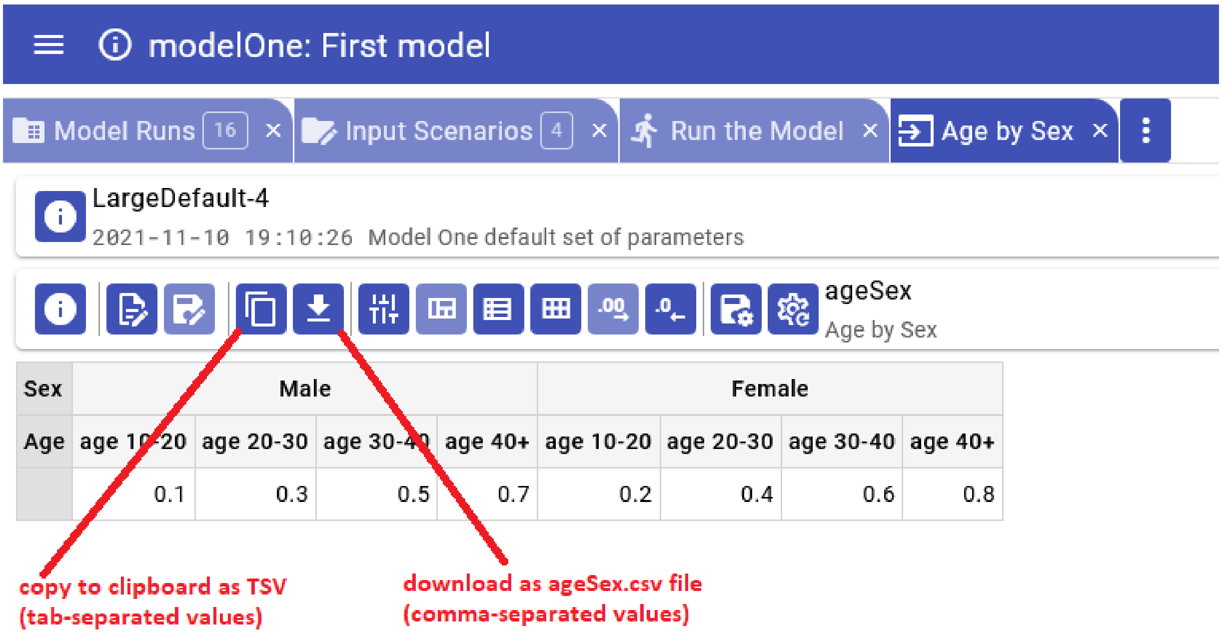
[back to topic contents]

Screenshot: Upload scenario



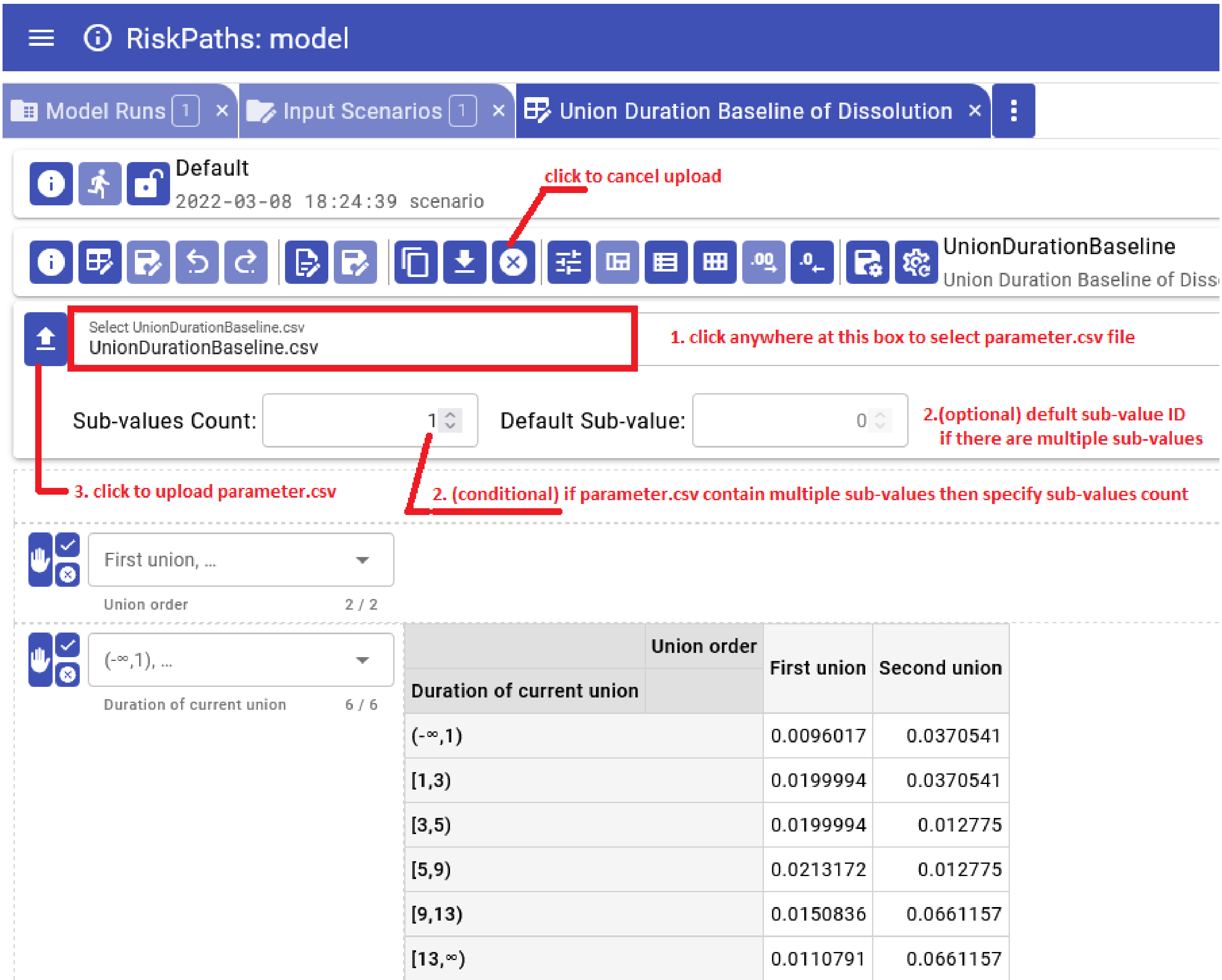
[back to topic contents]

Screenshot: Download parameter



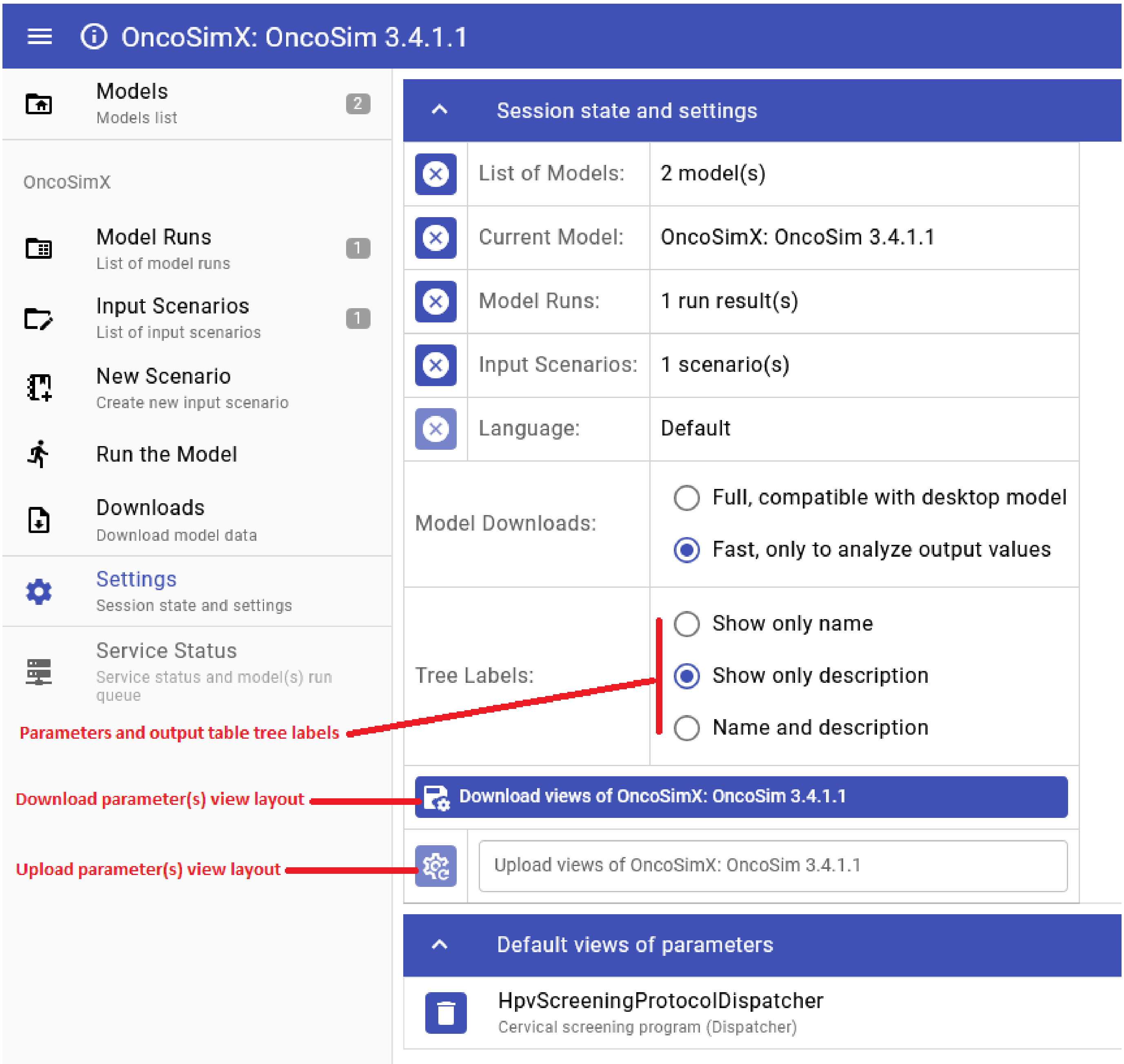
[back to topic contents]

Screenshot: Upload parameter



[back to topic contents]

Screenshot: Session state and settings



[back to topic contents]

UI: Create new or edit scenario

Home > **Create new scenario or edit existing scenario**

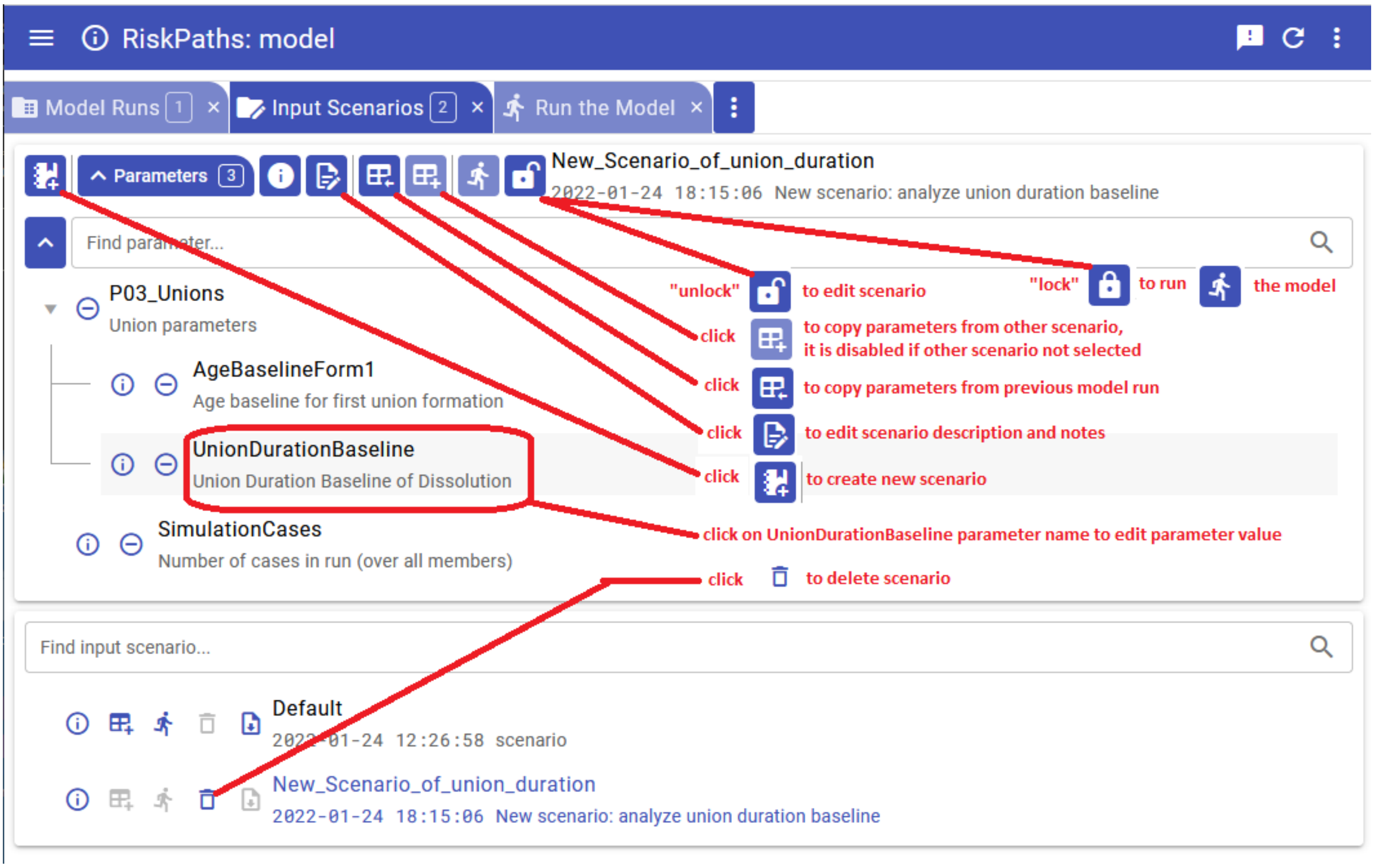
This topic shows functionality of the OpenM++ UI through annotated screenshots. The UI can also be explored by hovering over elements to display short descriptions.

User can do:

- edit parameter values using UI:
  - enter parameter values by typing or selecting from classification
  - copy-paste parameter values as TSV (tab separated values)
  - download parameter values as CSV file
  - upload parameter values as CSV file
- create new scenario:
  - enter new scenario description and notes
  - copy parameter or group of parameters from previous model run into the new scenario
  - copy parameter or group of parameters from other scenario into the new scenario
  - remove parameter or group of parameters from scenario
  - create new scenario and copy parameters which are different from base model run
- delete scenario
- edit existing scenario:
  - edit scenario description and notes
  - copy parameter or group of parameters from previous model run into the scenario
  - copy parameter or group of parameters from other scenario
  - remove parameter or group of parameters from scenario
- upload new scenario or upload new data to existing scenario

It is recommended to use "partial" scenario to run the model. Partial scenarios contain only parameters which you want to modify for your analysis. For example, for `RiskPaths` model it can be only number of `Simulation Cases` and `Union Duration Baseline`. All other parameters, which you don't want to change, can come either from previous model run, or from Default model scenario.

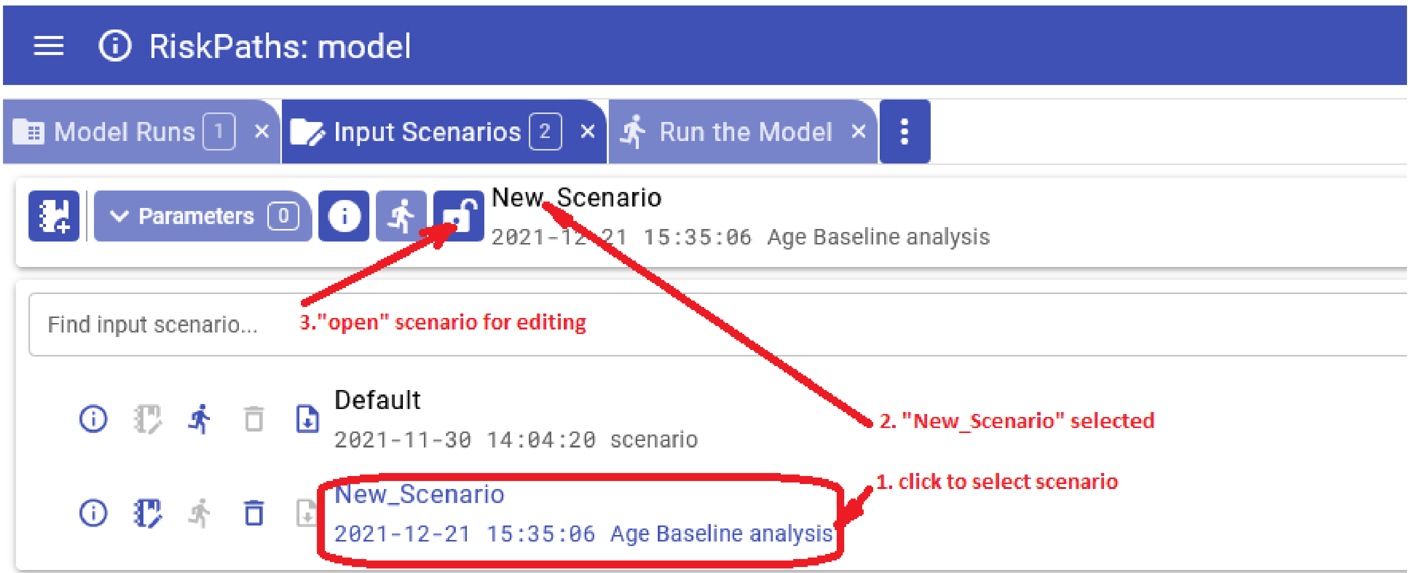
To create new scenario or to modify existing scenario click on Input Scenarios tab:



To edit existing scenario or to modify parameter(s) do:

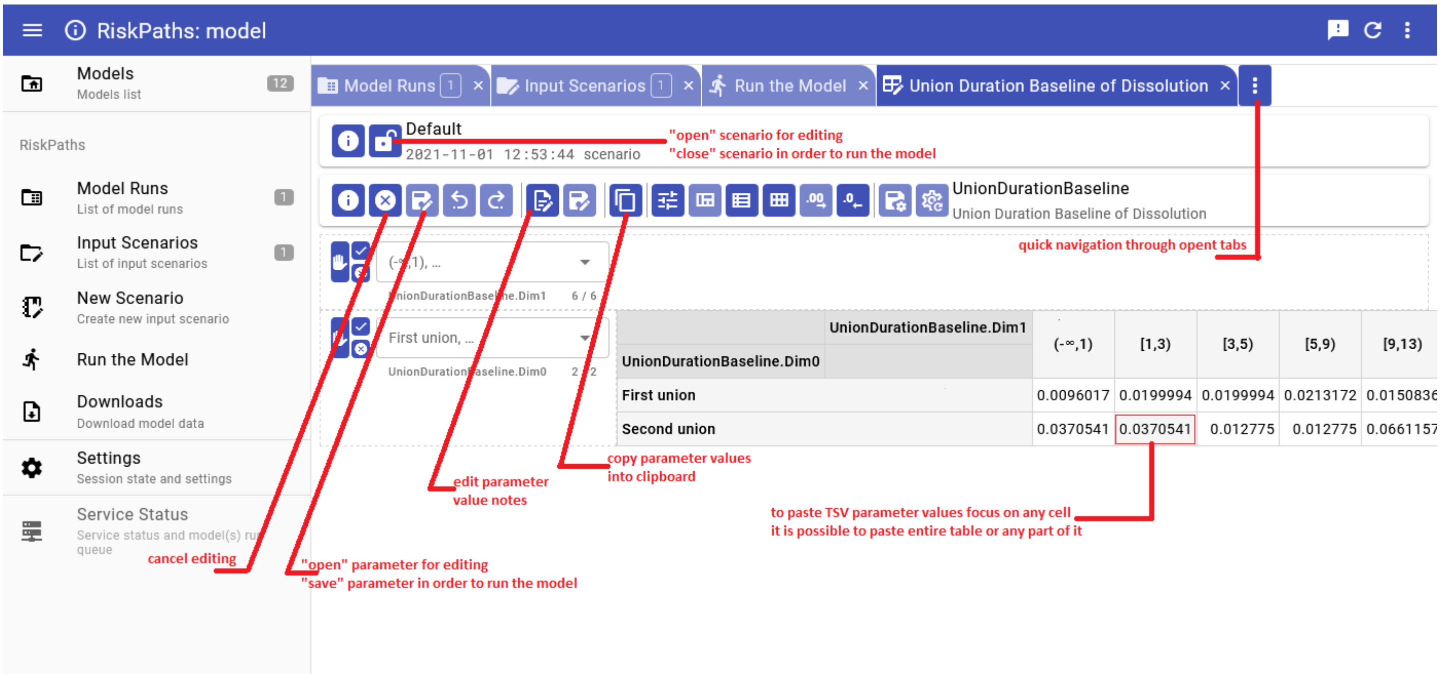
- select scenario from the list
- if scenario is "locked" then click on "unlock button"

Imporatnt: After scenario editing completed click on "lock" button to use that scenario for model run. Scenario must be "locked" in order to be runnable, you can NOT run "unlocked" scenario.

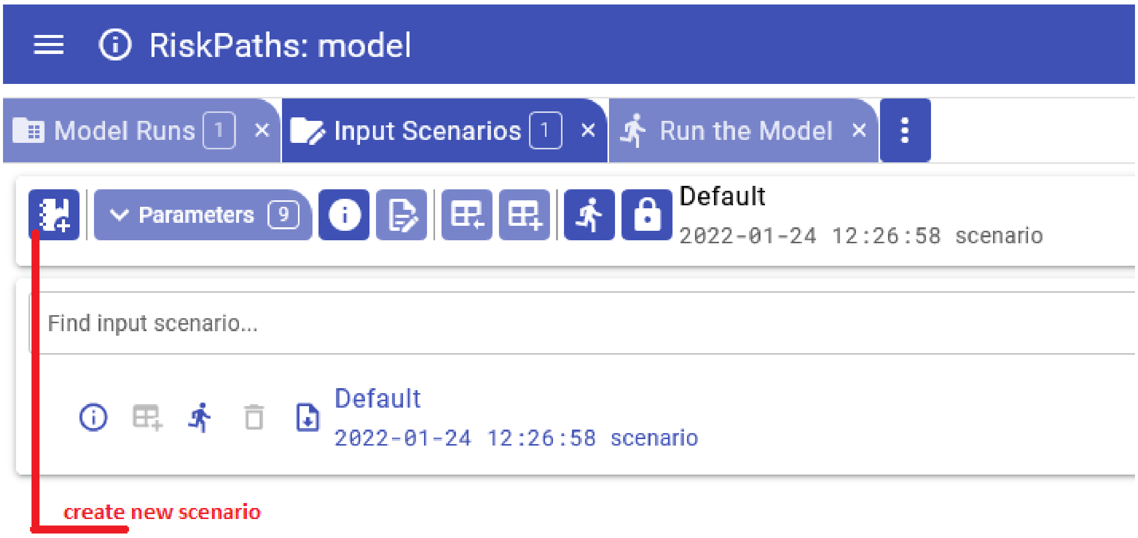


In order to open parameter values editor click on parameter name in the scenario parameters tree. For example, click on `UnionDurationBaseline` parameter of `RiskPaths` model.

Please keep in mind, in openM++ number of Simulation Cases is also a model parameter (it is a different from Modgen).

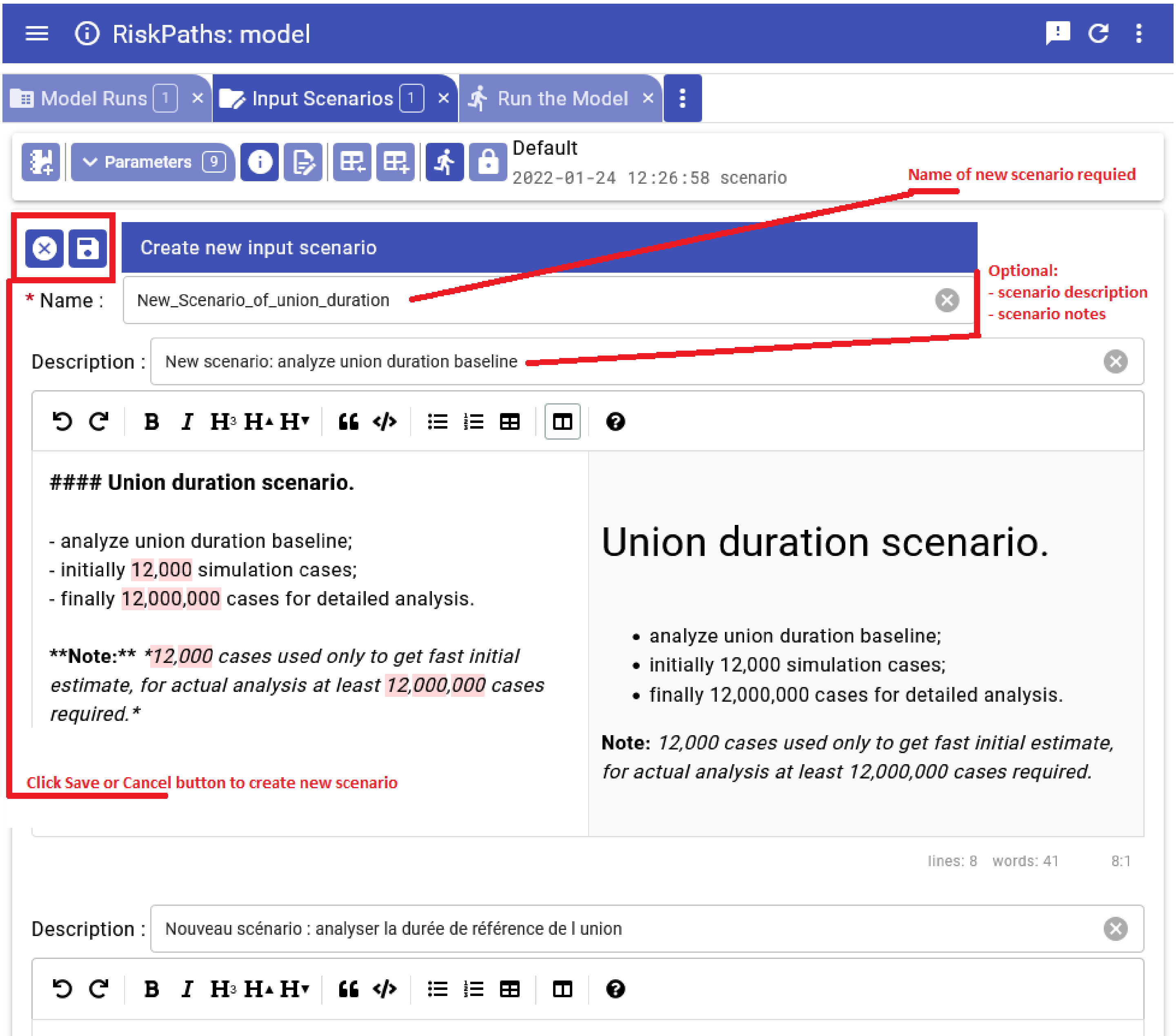


To create new scenario click on new scenario button:



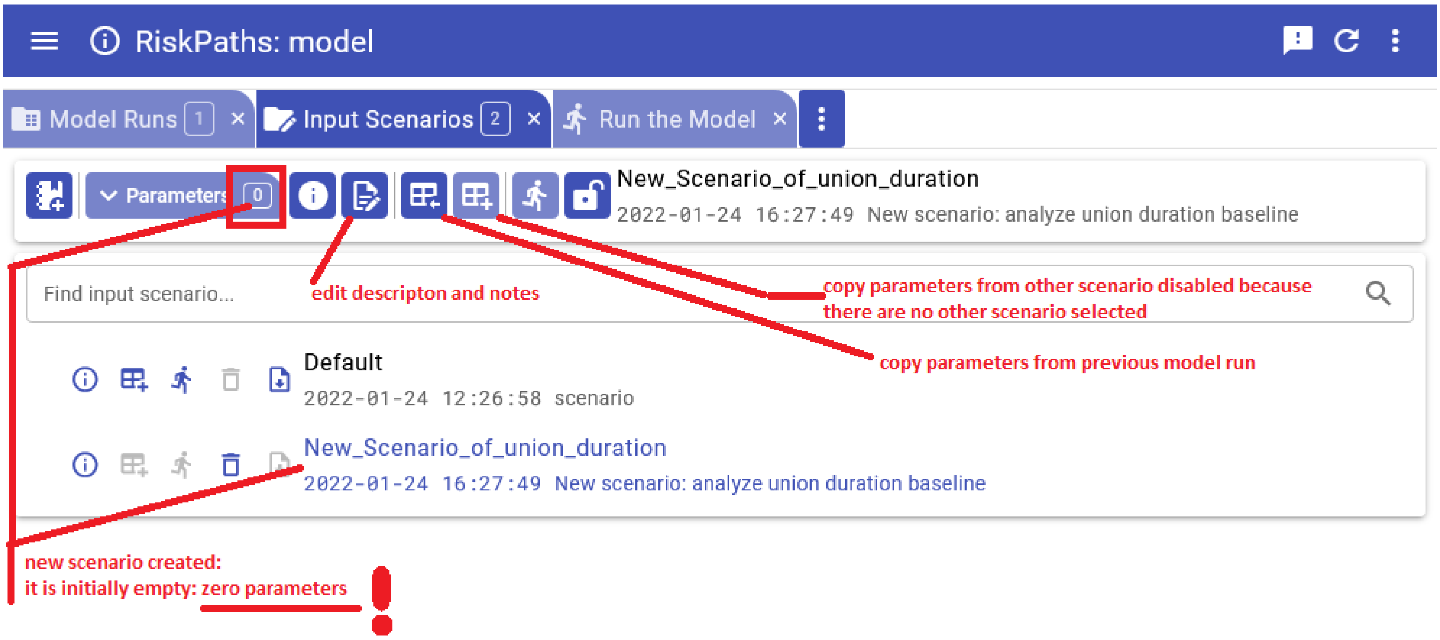
Create new scenario:

- provide new scenario name. It must be a valid file name, and cannot contain any of: `"'`:*?><|$}{@&^;/\`
- you cannot change scenario name later, there is no "rename" scenario option;
- (optional) provide scenario description and notes, you always can change description and notes later.
- click on Save button to save scenario or on Cancel to discard your changes



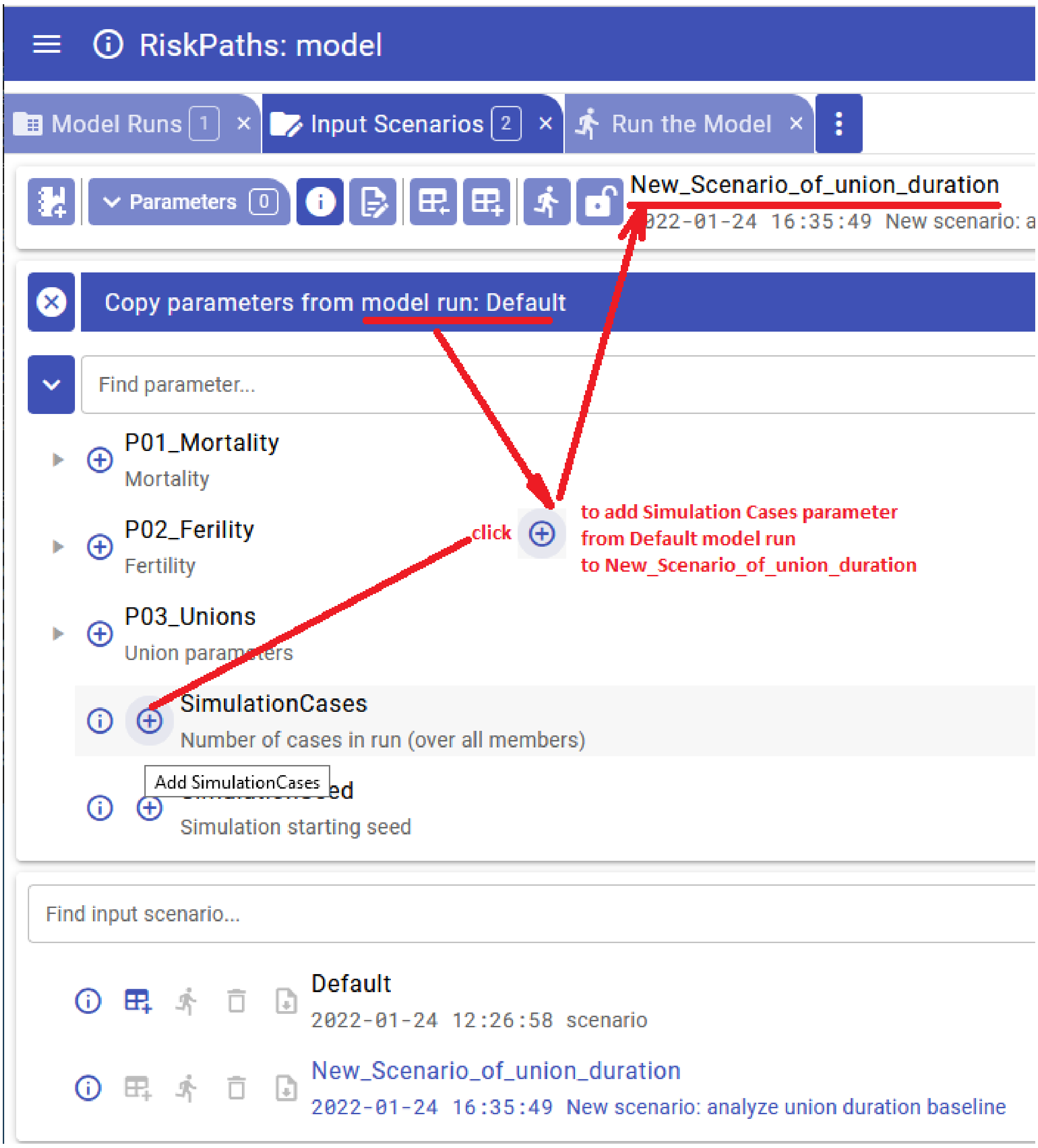
After new scenario created you can add parameters into it by copy it:

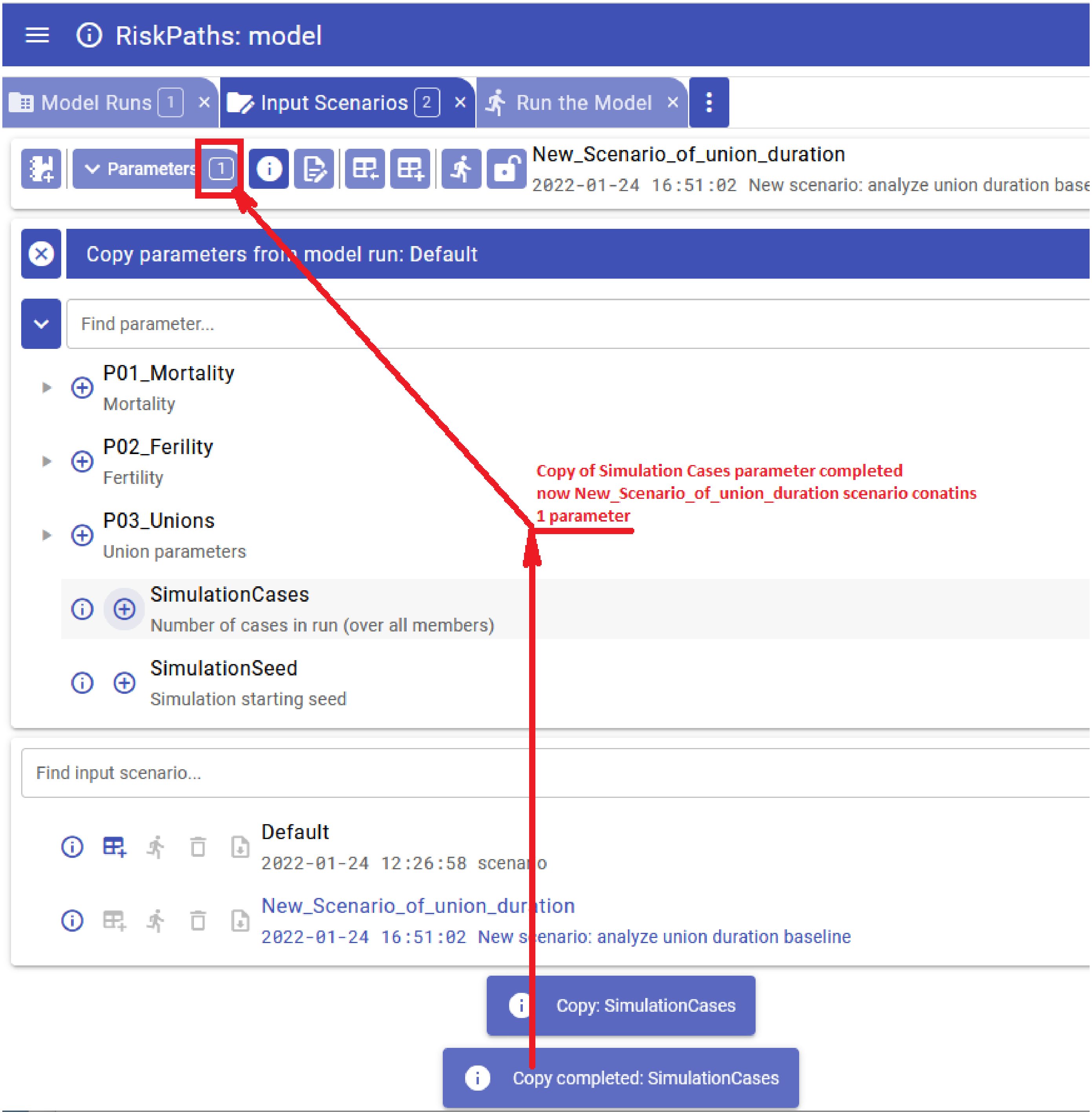
- from previous model run
- or from other scenario



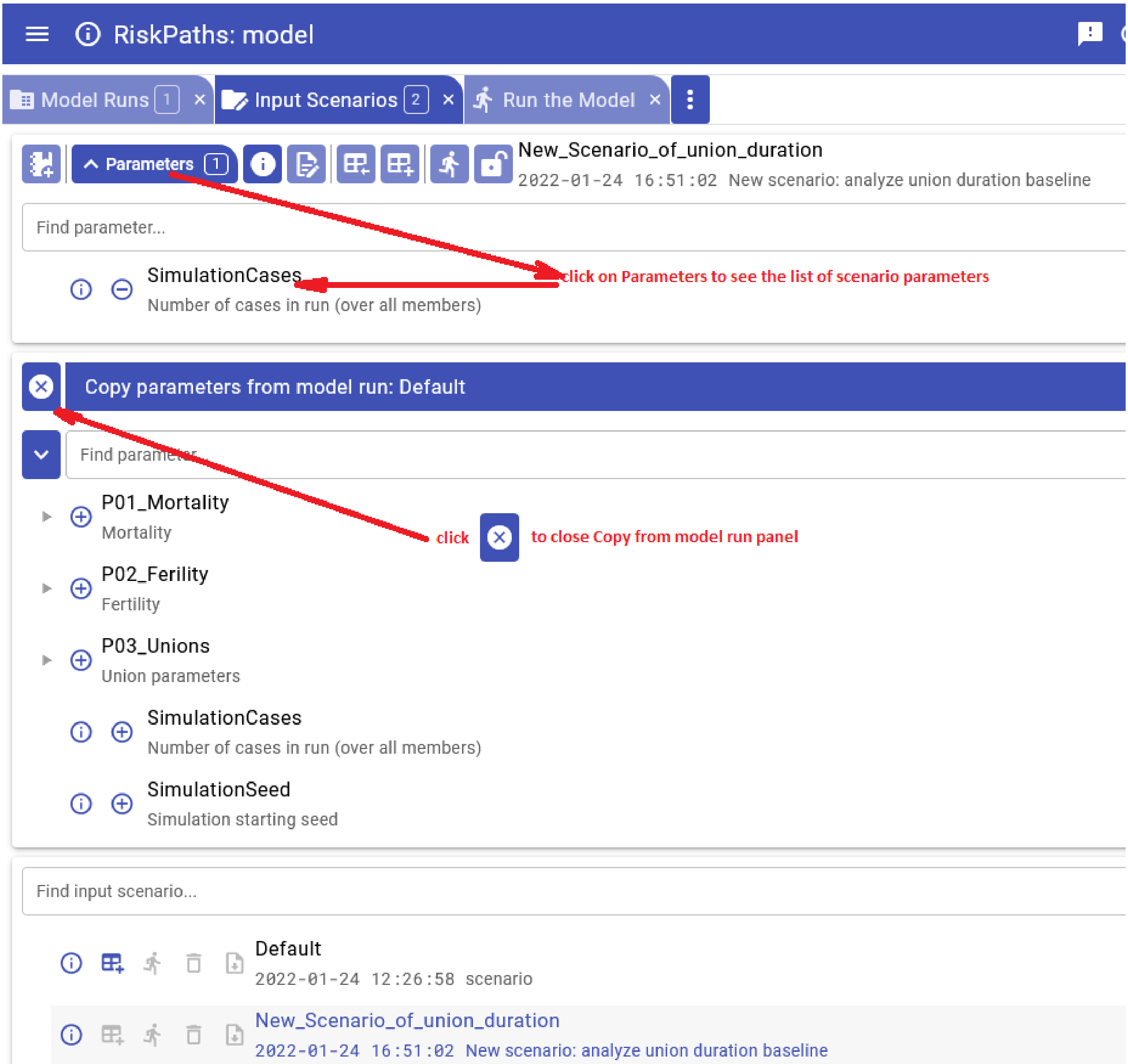
In order to copy parameter(s) from previous model run:

- click on "Copy from previous model run" button (see above)
- select parameter from model parameters tree
- click on plus button


RiskPaths: model
Model Runs 1
Input Scenarios 2
Run the Model
Parameters 0
New_Scenario_of_union_duration
2022-01-24 16:35:49 New scenario: a
Copy parameters from model run: Default
Find parameter...
P01_Mortality
Mortality
P02_Ferility
Fertility
click
to add Simulation Cases parameter
from Default model run
to New_Scenario_of_union_duration
P03_Unions
Union parameters
SimulationCases
Number of cases in run (over all members)
Add SimulationCases
Simulation starting seed
Find input scenario...
Default
2022-01-24 12:26:58 scenario
New_Scenario_of_union_duration
2022-01-24 16:35:49 New scenario: analyze union duration baseline

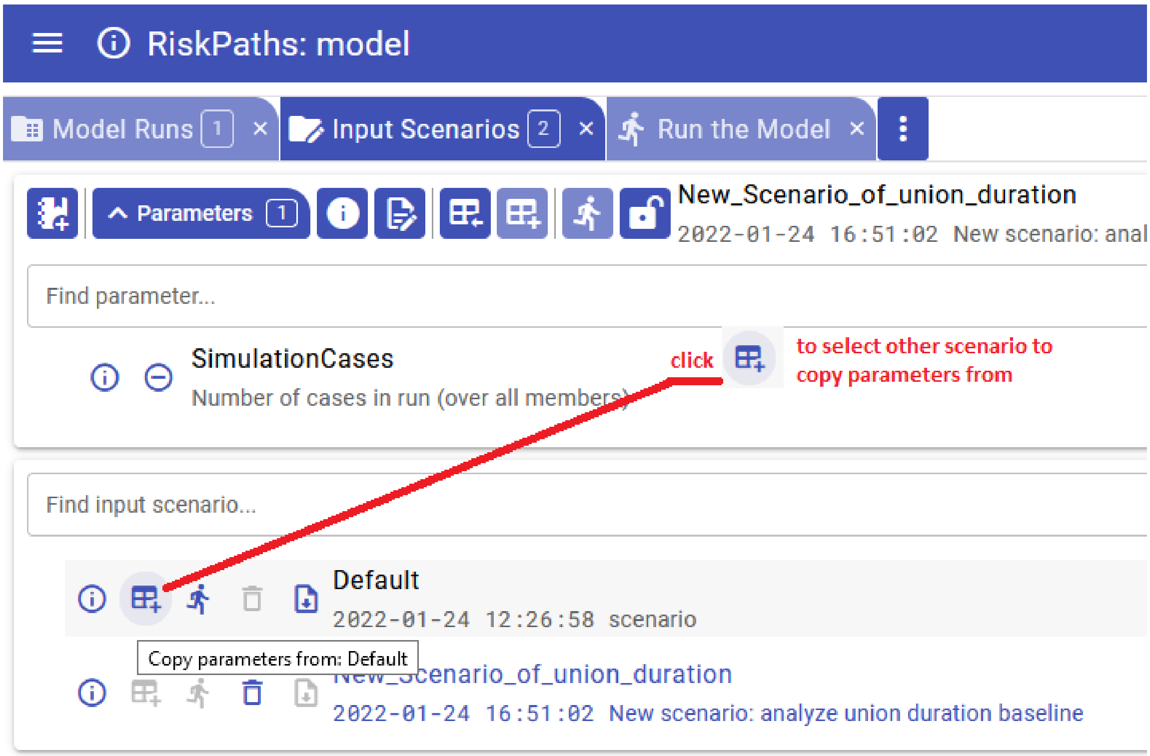


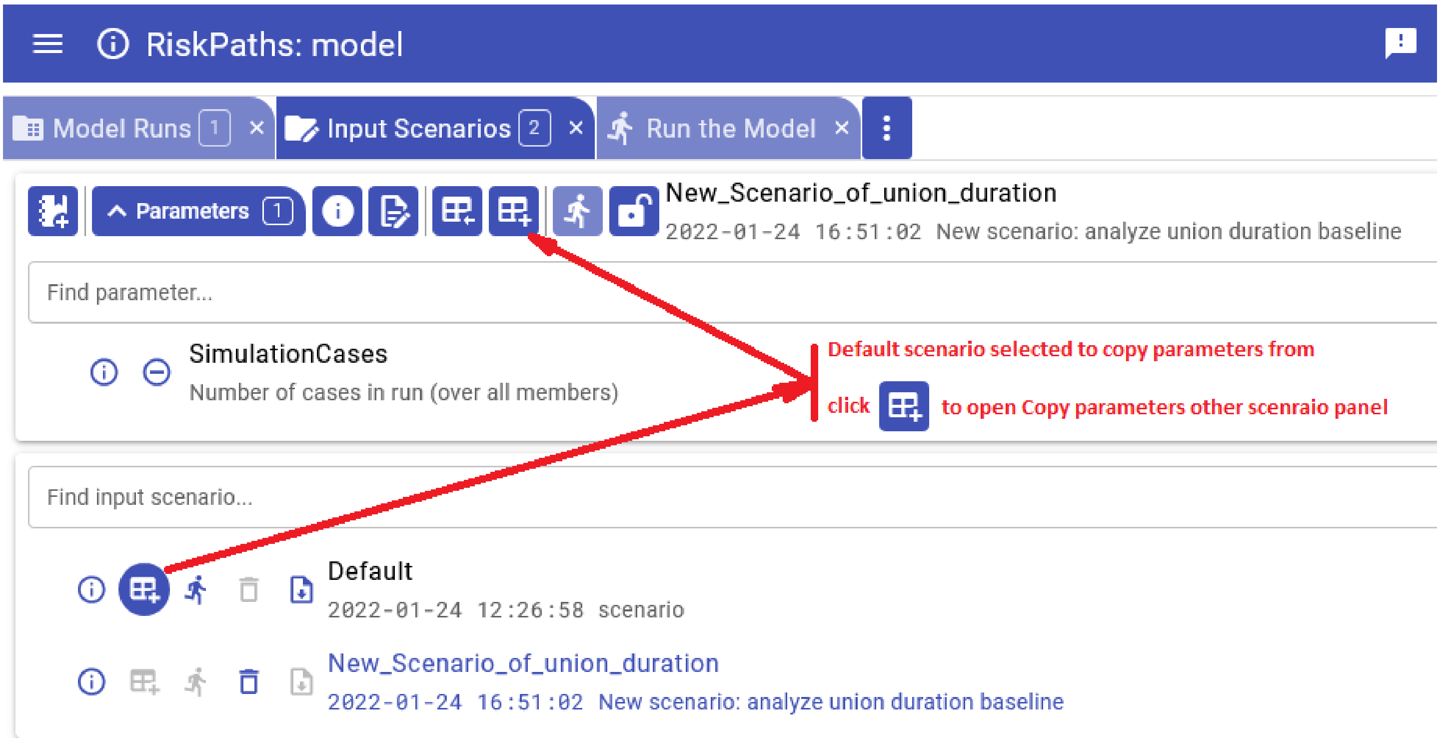
After you are done with copy parameters from previous model click on Close button:



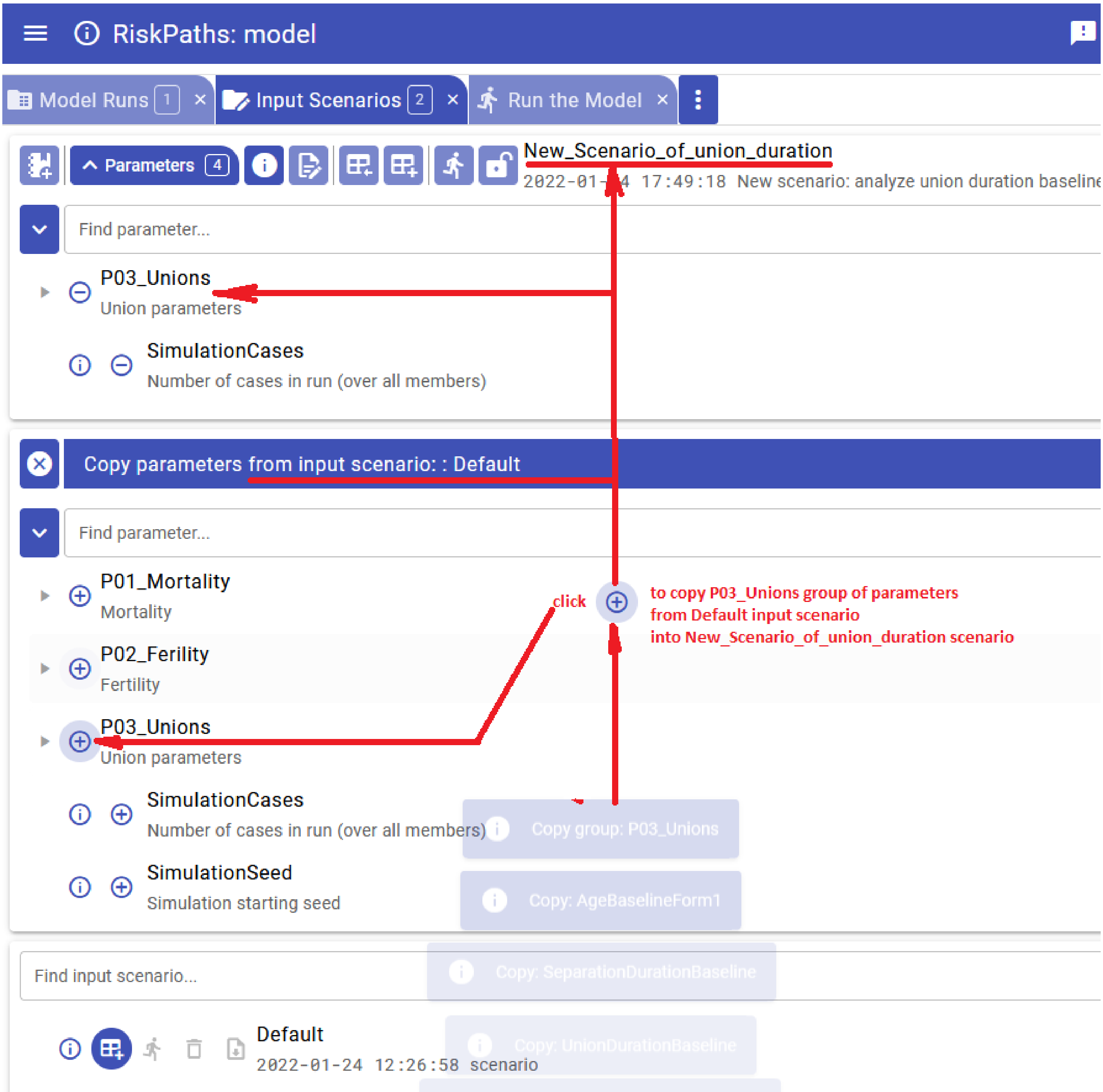
In order to copy parameter(s) from other input scenario:

- select source input scenario from the list. Source scenario must be "locked" otherwise you would not be able to select it as a source of parameters
- click on "Copy from other scenario" button (see below)



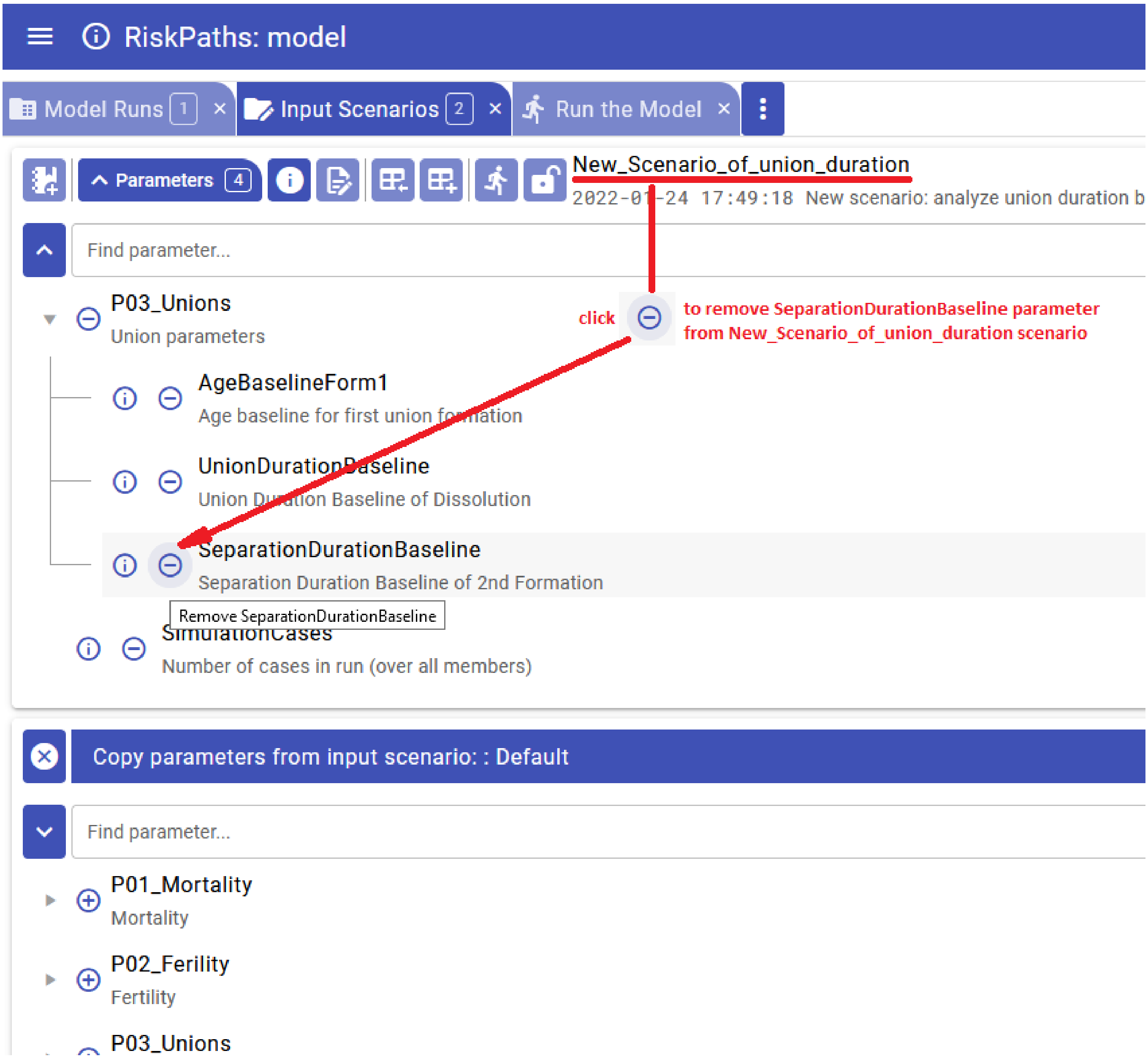


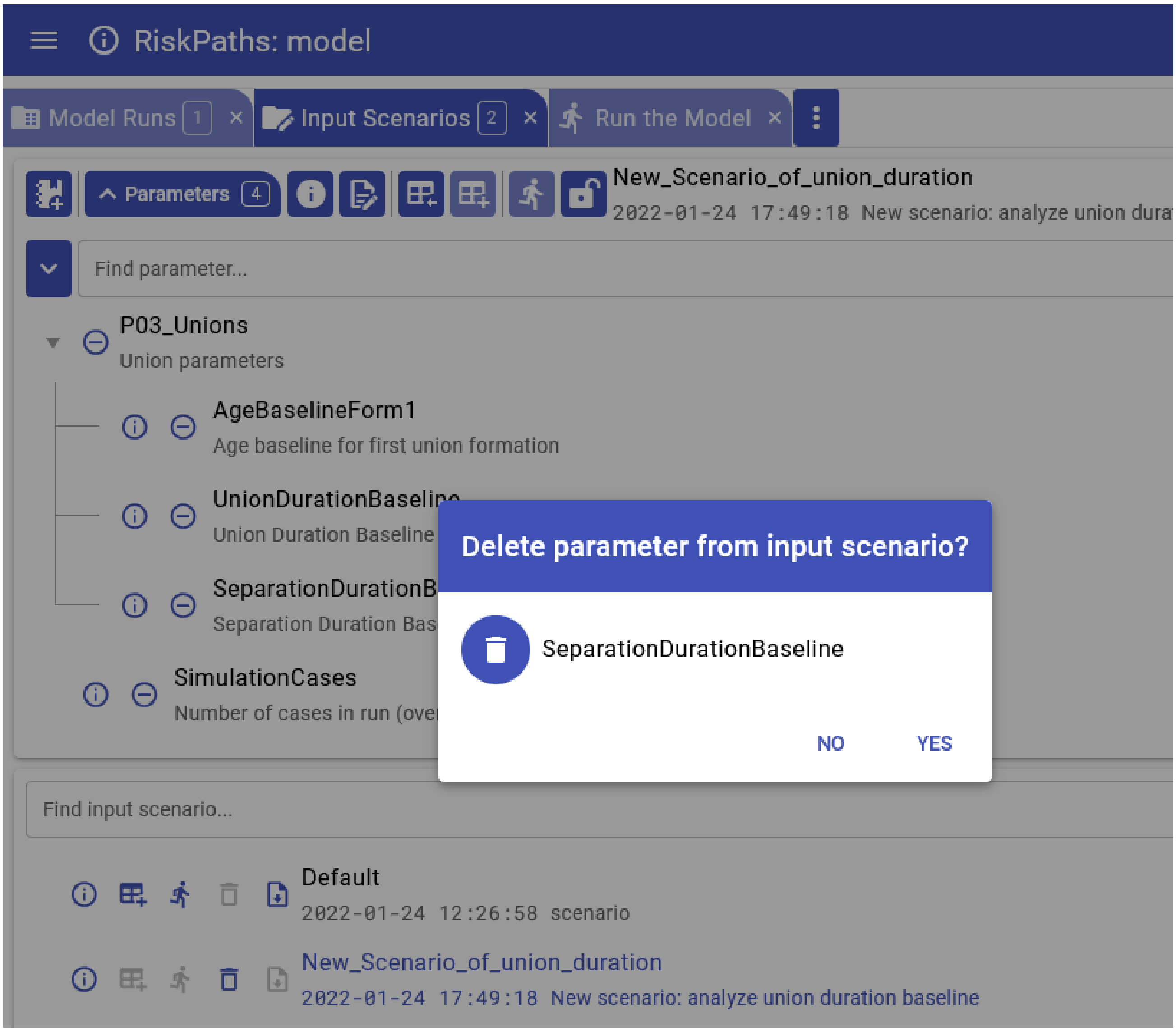
To copy parameter or group of parameters click on plus button in the scenario parameters tree



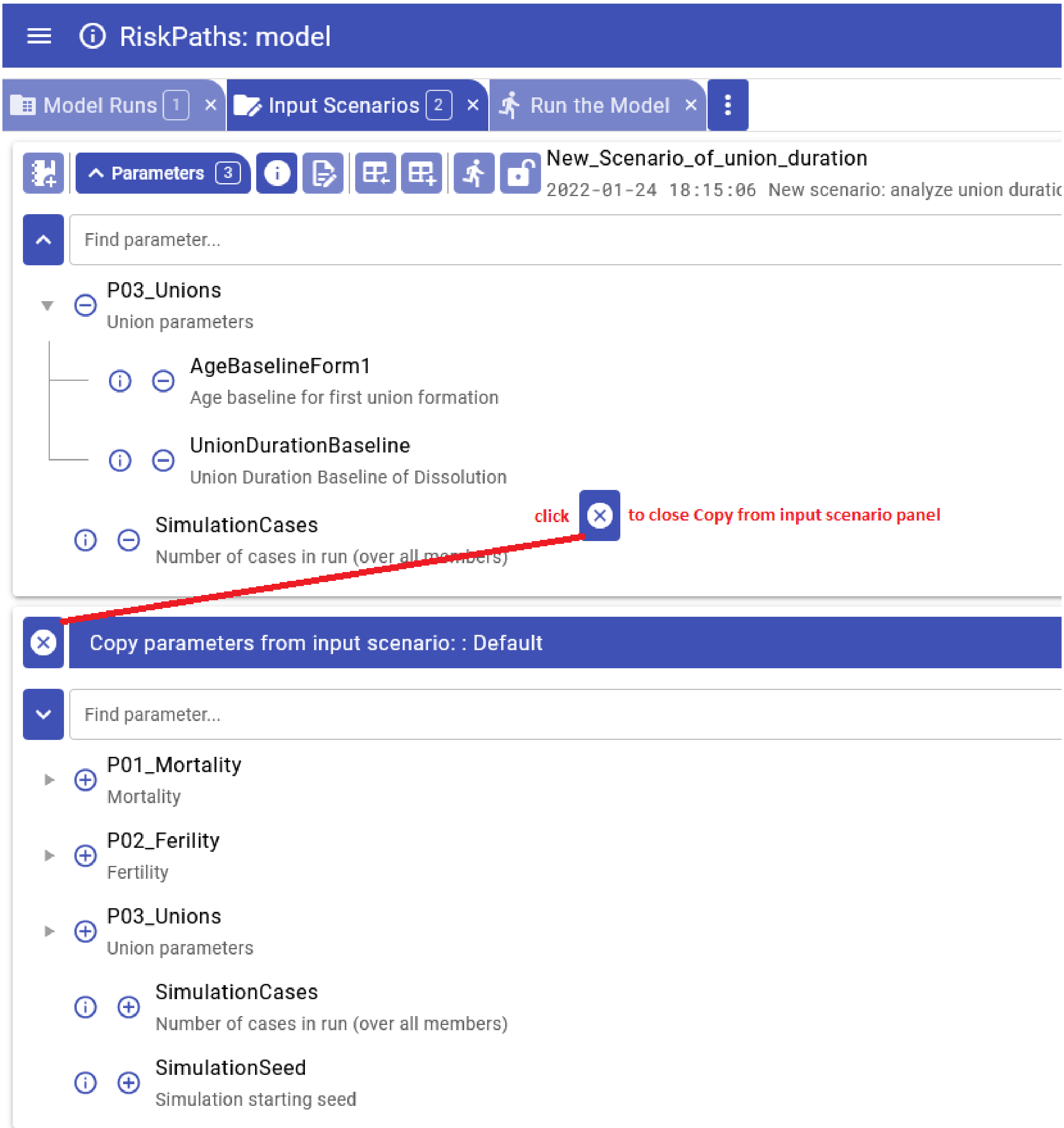
In order to delete parameter from your current scenario:

- click on minus button in the scenario parameters tree:
- confirm "Yes" to remove parameter values from scenario.

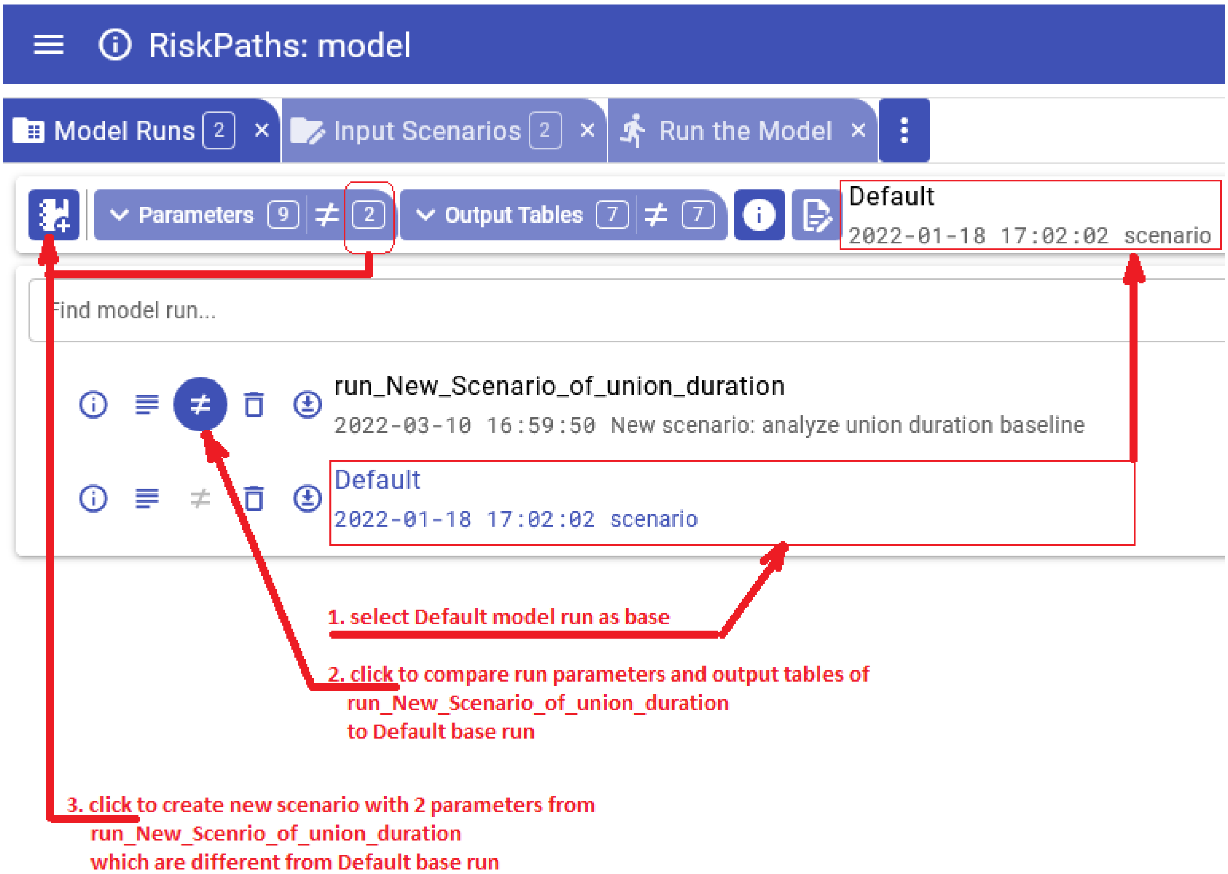

RiskPaths: model
Model Runs 1
Input Scenarios 2
Run the Model
Parameters 4
New_Scenario_of_union_duration
2022-01-24 17:49:18 New scenario: analyze union duration b
Find parameter...
P03_Unions
Union parameters
click
to remove SeparationDurationBaseline parameter
from New_Scenario_of_union_duration scenario
AgeBaselineForm1
Age baseline for first union formation
UnionDurationBaseline
Union Duration Baseline of Dissolution
SeparationDurationBaseline
Separation Duration Baseline of 2nd Formation
Remove SeparationDurationBaseline
SimulationCases
Number of cases in run (over all members)
Copy parameters from input scenario: : Default
Find parameter...
P01_Mortality
Mortality
P02_Ferility
Fertility
P03_Unions



After you are done with copy parameters from other scenario click on Close button:



User can create new scenario from results of run comparison. In that case scenario will include all parameters of that model run which are different from the base model run.


RiskPaths: model
Model Runs 2
Input Scenarios 2
Run the Model
Parameters 9 ≠ 2
Output Tables 7 ≠ 7
Default
2022-01-18 17:02:02 scenario
Find model run...
run_New_Scenario_of_union_duration
2022-03-10 16:59:50 New scenario: analyze union duration baseline
Default
2022-01-18 17:02:02 scenario
1. select Default model run as base
2. click to compare run parameters and output tables of
run_New_Scenario_of_union_duration
to Default base run
3. click to create new scenario with 2 parameters from
run_New_Scenrio_of_union_duration
which are different from Default base run

UI: Upload input scenario or parameters

This topic shows functionality of the OpenM++ UI through annotated screenshots. The UI can also be explored by hovering over elements to display short descriptions.

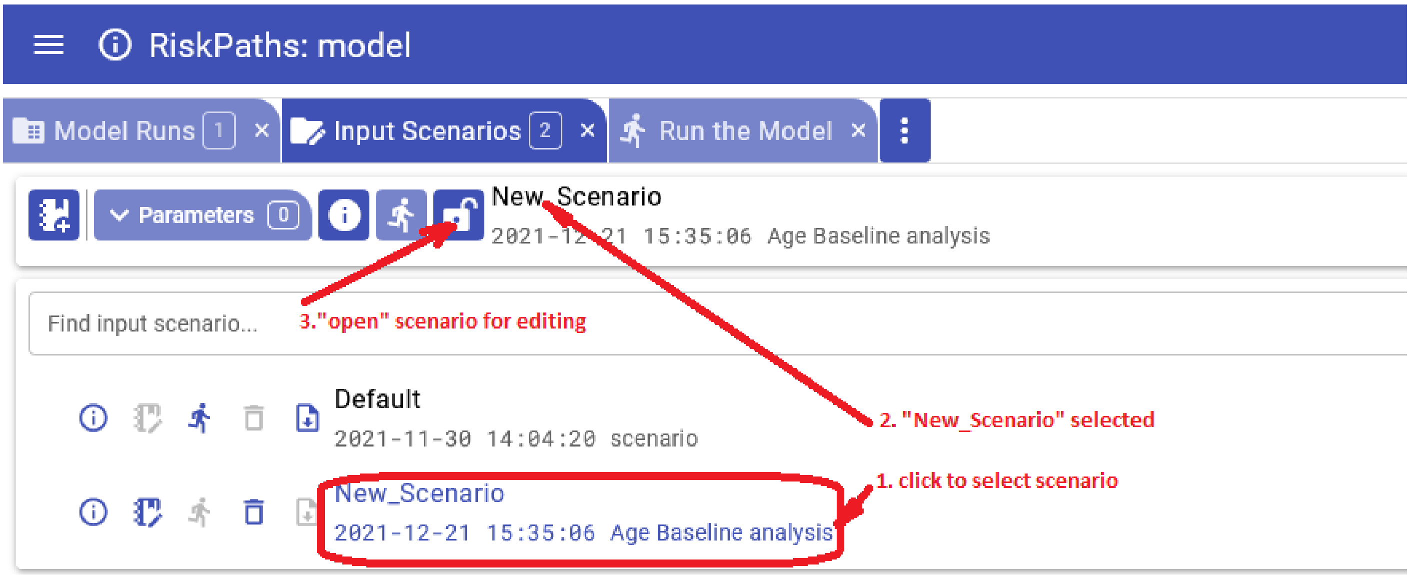
User can modify existing or create new input scenario by uploading `scenario.zip` archive. Such archive can be created by `dbcopy` utility or through UI download or by Create Import Set utility. In most simplest case such ZIP archive can contain one or more CSV parameter file(s).

User also can replace existing parameter values by uploading parameter CSV file.

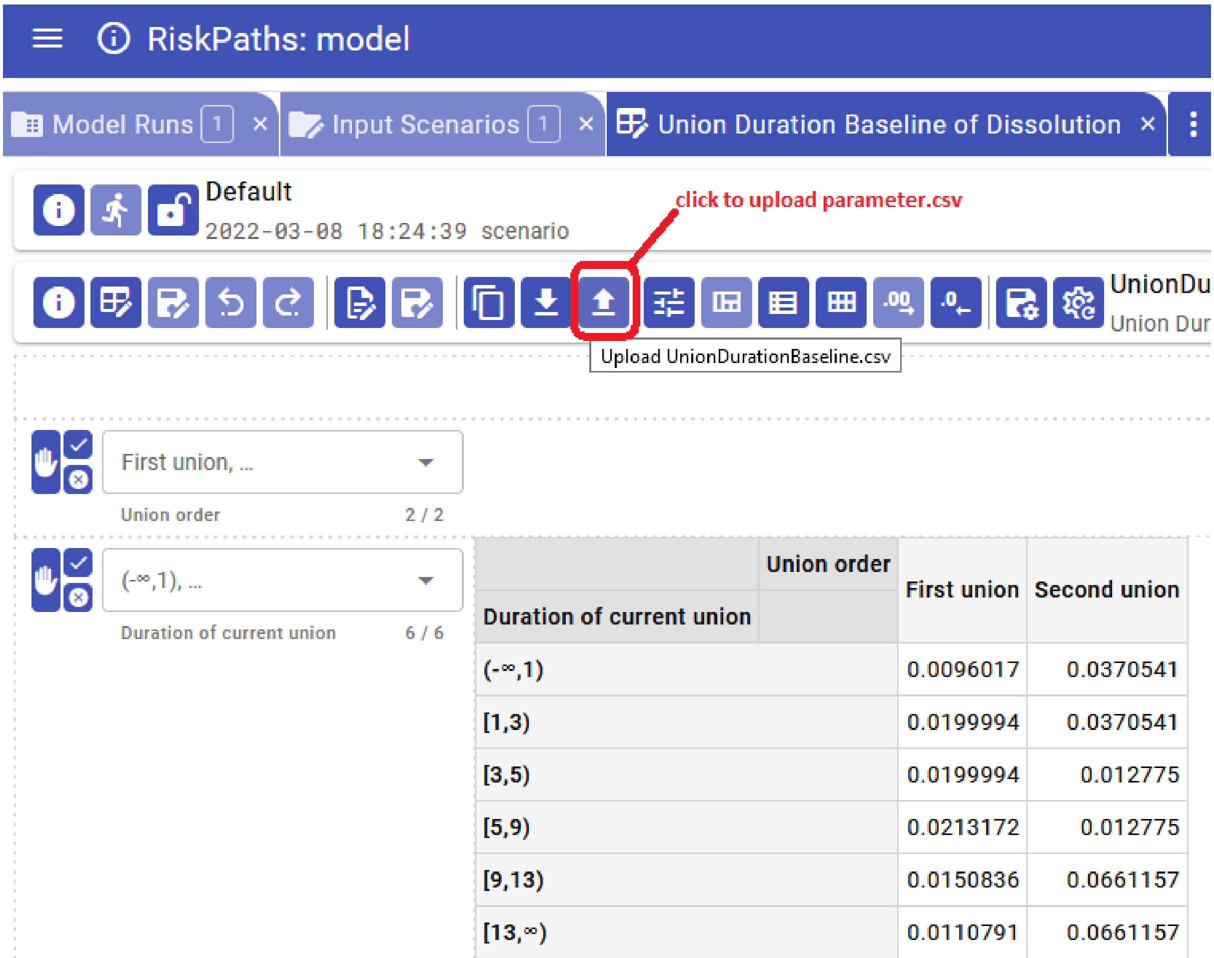
To edit existing scenario or to modify parameter(s) do:

- select scenario from the list
- if scenario is "locked" then click on "unlock button"

Imporatnt: After scenario editing completed click on "lock" button to use that scenario for model run. Scenario must be "locked" in order to be runnable, you can NOT run "unlocked" scenario.



To replace existing parameter values by uploading parameter CSV file click on Upload parameter button:

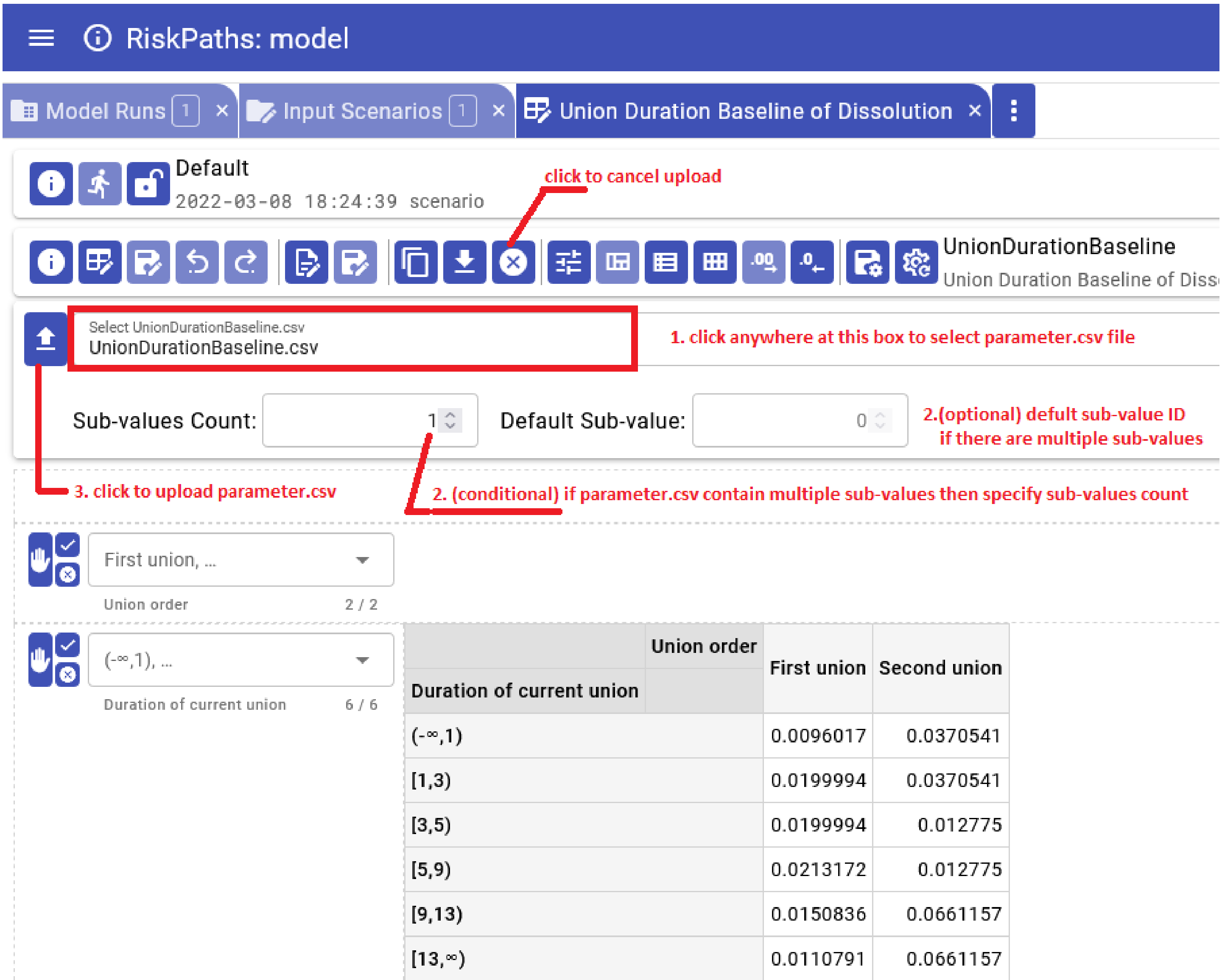


Parameter.csv files described at:

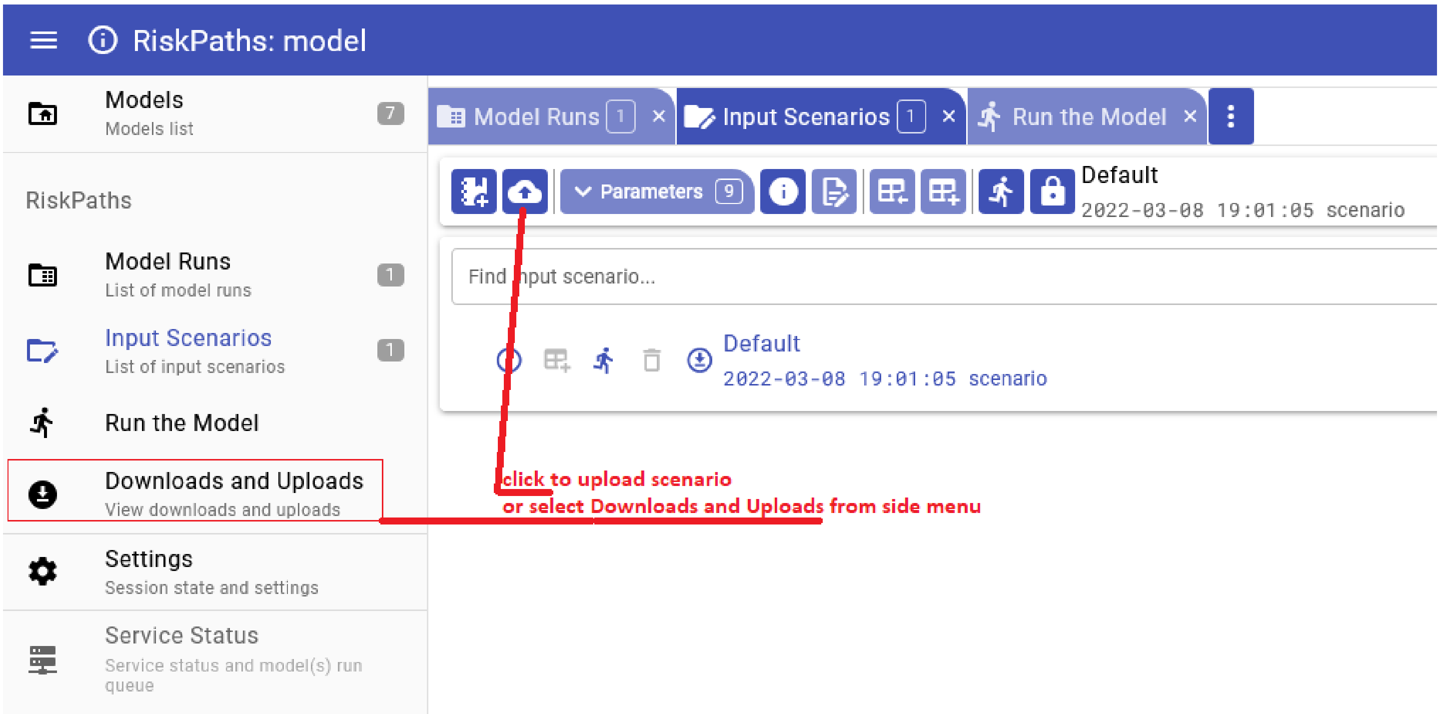
- How To Set Model Parameters
- Model Run: How model finds input parameters

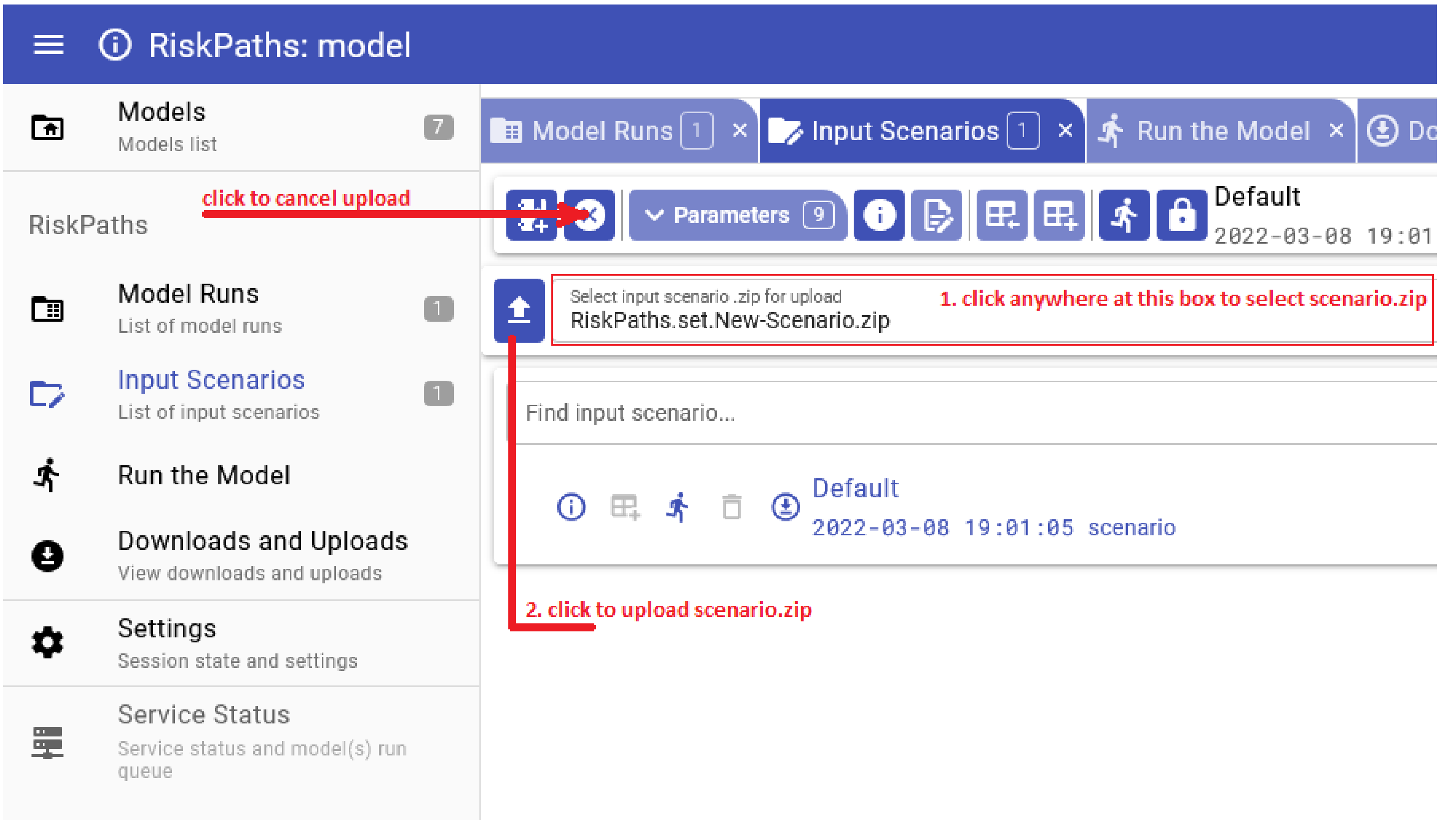
It is possible to supply multiple sub-values inside of parameter.csv file, in that case:

- user must specify number of sub-values in the input CSV file
- user can specify default sub-value ID for that parameter, if it is not a zero.

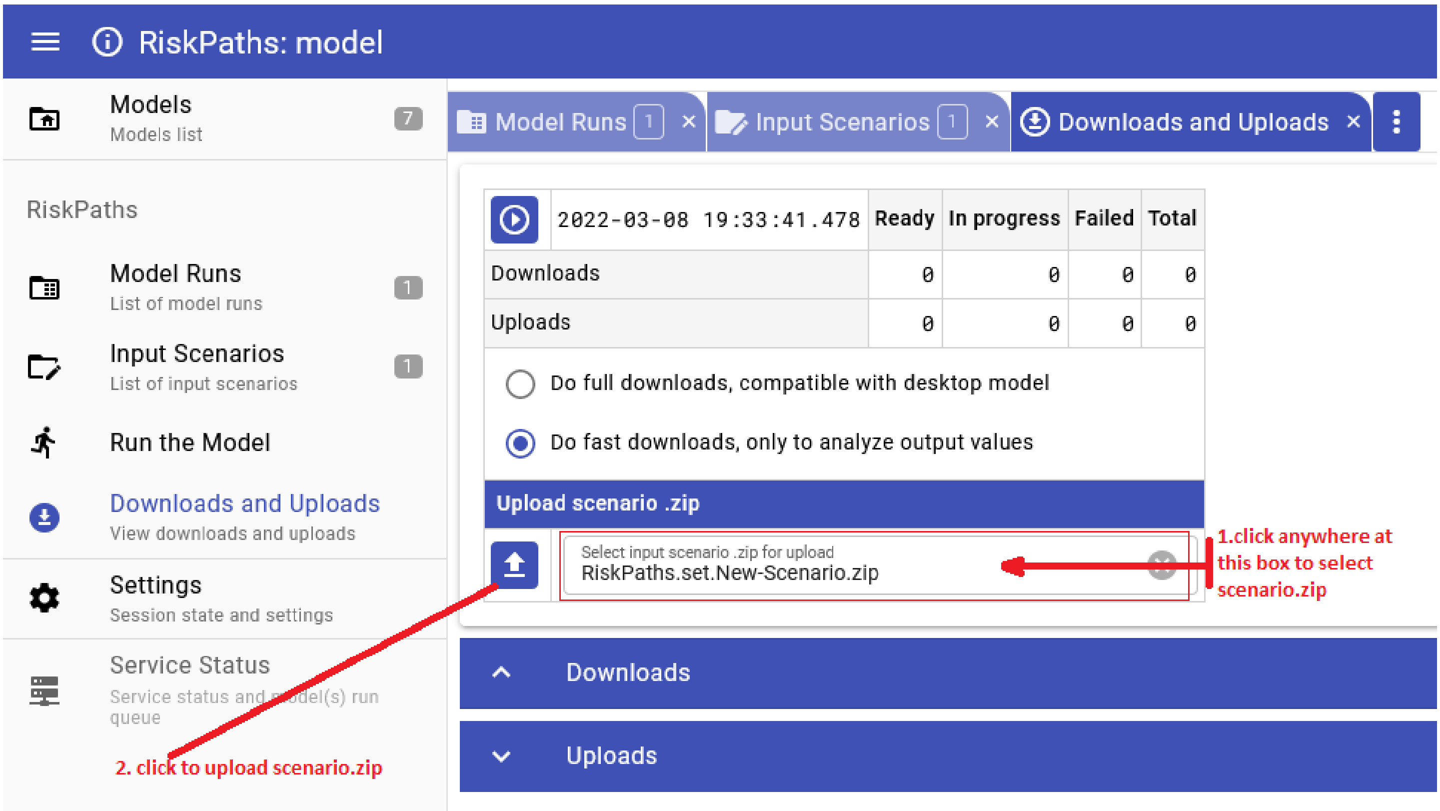


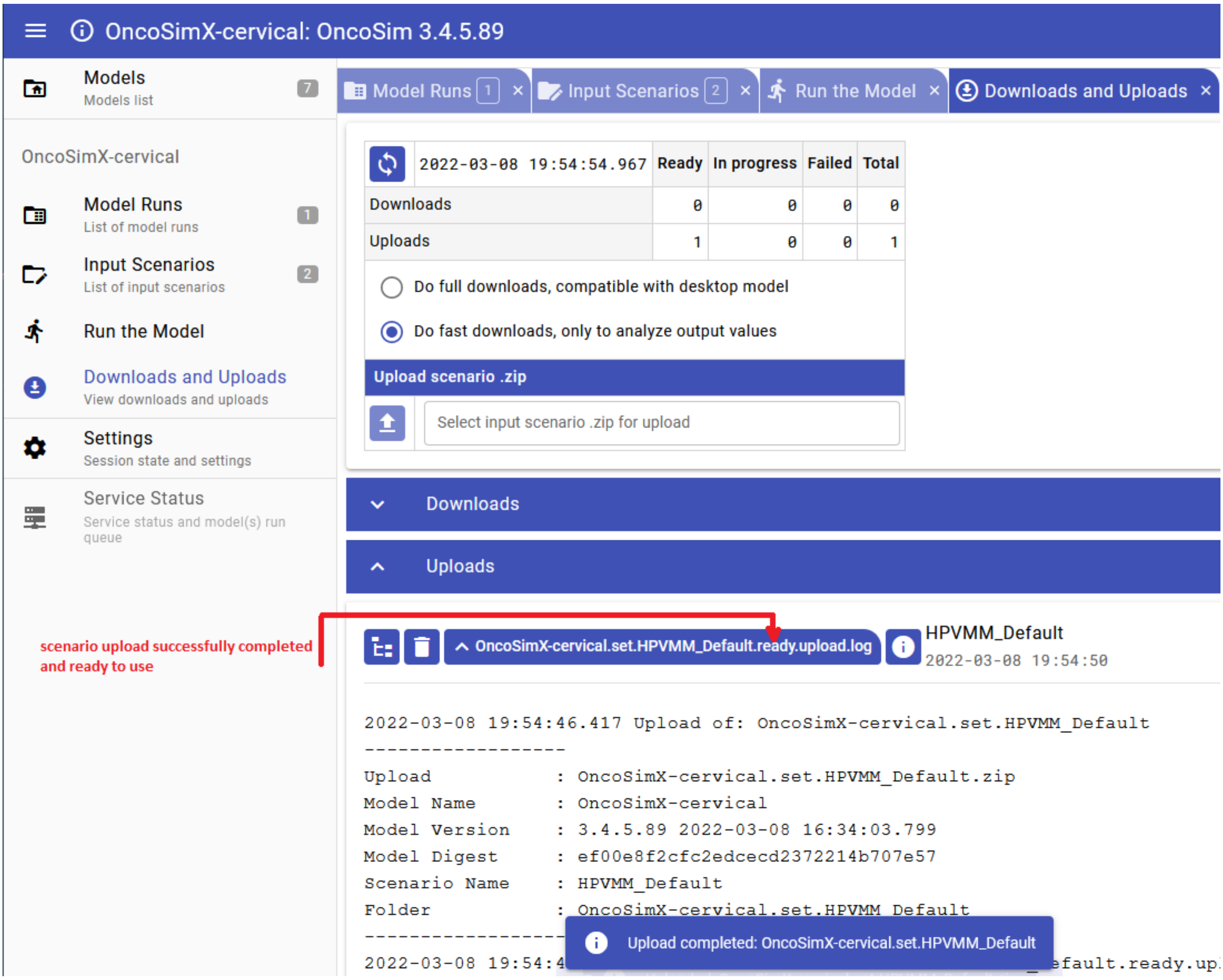
User can upload scenario ZIP archive from input scenarios list page:





It is also possible to do upload from Downloads and Uploads page:




OncoSimX-cervical: OncoSim 3.4.5.89
Models
Models list
7
Model Runs 1
Input Scenarios 2
Run the Model
Downloads and Uploads
OncoSimX-cervical
Model Runs
List of model runs
1
Input Scenarios
List of input scenarios
2
Run the Model
Downloads and Uploads
View downloads and uploads
Settings
Session state and settings
Service Status
Service status and model(s) run queue
2022-03-08 19:54:54.967
Ready
In progress
Failed
Total
Downloads 0 0 0 0
Uploads 1 0 0 1
Do full downloads, compatible with desktop model
Do fast downloads, only to analyze output values
Upload scenario .zip
Select input scenario .zip for upload
Downloads
Uploads
scenario upload successfully completed and ready to use
OncoSimX-cervical.set.HPVMM_Default.ready.upload.log
HPVMM_Default
2022-03-08 19:54:50
2022-03-08 19:54:46.417 Upload of: OncoSimX-cervical.set.HPVMM_Default
Upload : OncoSimX-cervical.set.HPVMM_Default.zip
Model Name : OncoSimX-cervical
Model Version : 3.4.5.89 2022-03-08 16:34:03.799
Model Digest : ef00e8f2cfc2edcecd2372214b707e57
Scenario Name : HPVMM_Default
Folder : OncoSimX-cervical.set.HPVMM_Default
Upload completed: OncoSimX-cervical.set.HPVMM_Default

UI: Run the Model

Home > **Run the Model**

This topic shows functionality of the OpenM++ UI through annotated screenshots. The UI can also be explored by hovering over elements to display short descriptions.

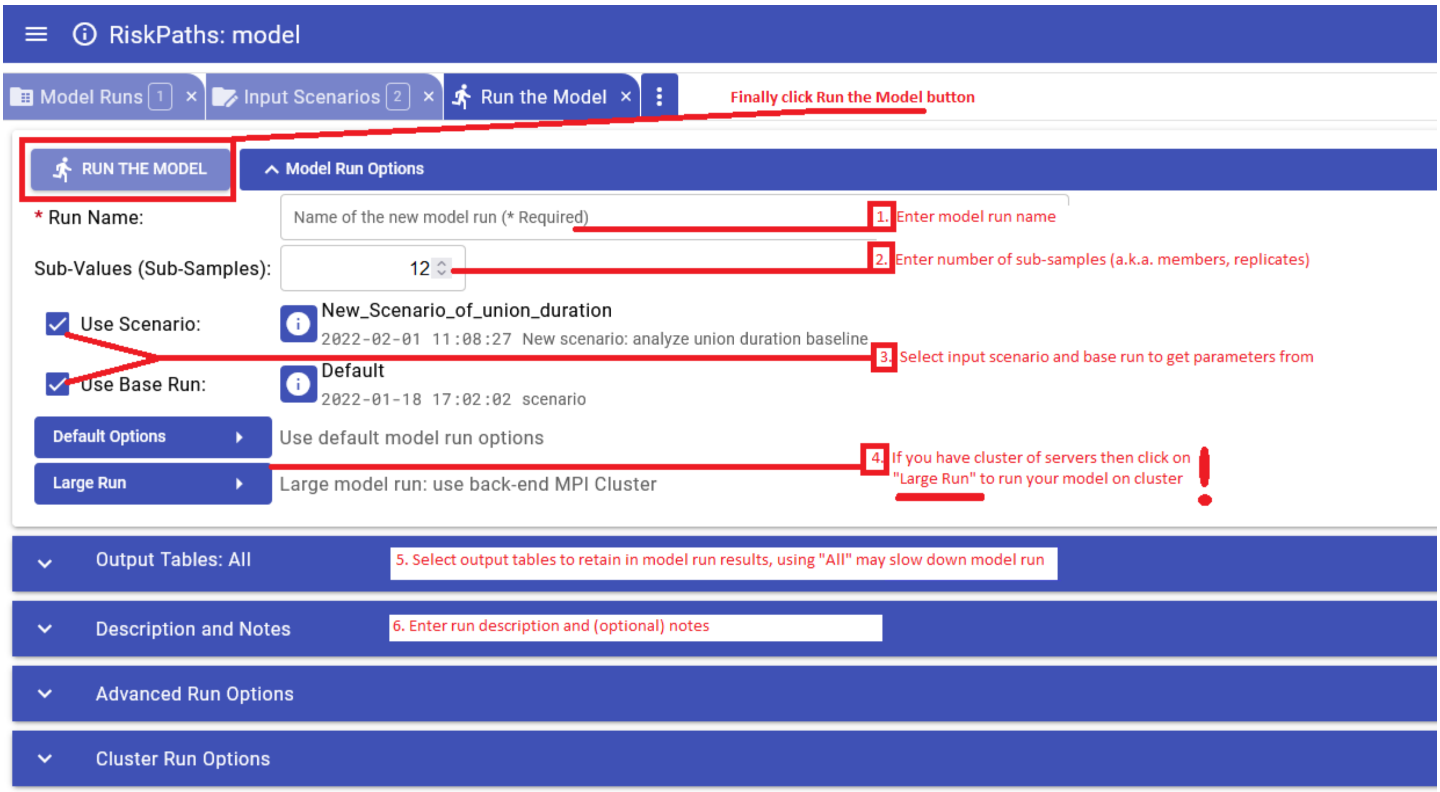
It is recommended to use "partial" scenario to run the model. Partial scenarios contain only parameters which you want to modify for your analysis. For example, for `RiskPaths` model it can be only number of `Simulation Cases` and `Union Duration Baseline`. All other parameters, which you don't want to change, can come from previous model run (a.k.a. Base Run).

Sub-values: sub-samples, members, replicas: *Following terms: "simulation member", "replicate", "sub-sample" are often used in micro-simulation conversations interchangeably, depending on context. To avoid terminology discussion openM++ uses "sub-value" as equivalent of all above and it is the same as "sub-sample" in Modgen.*

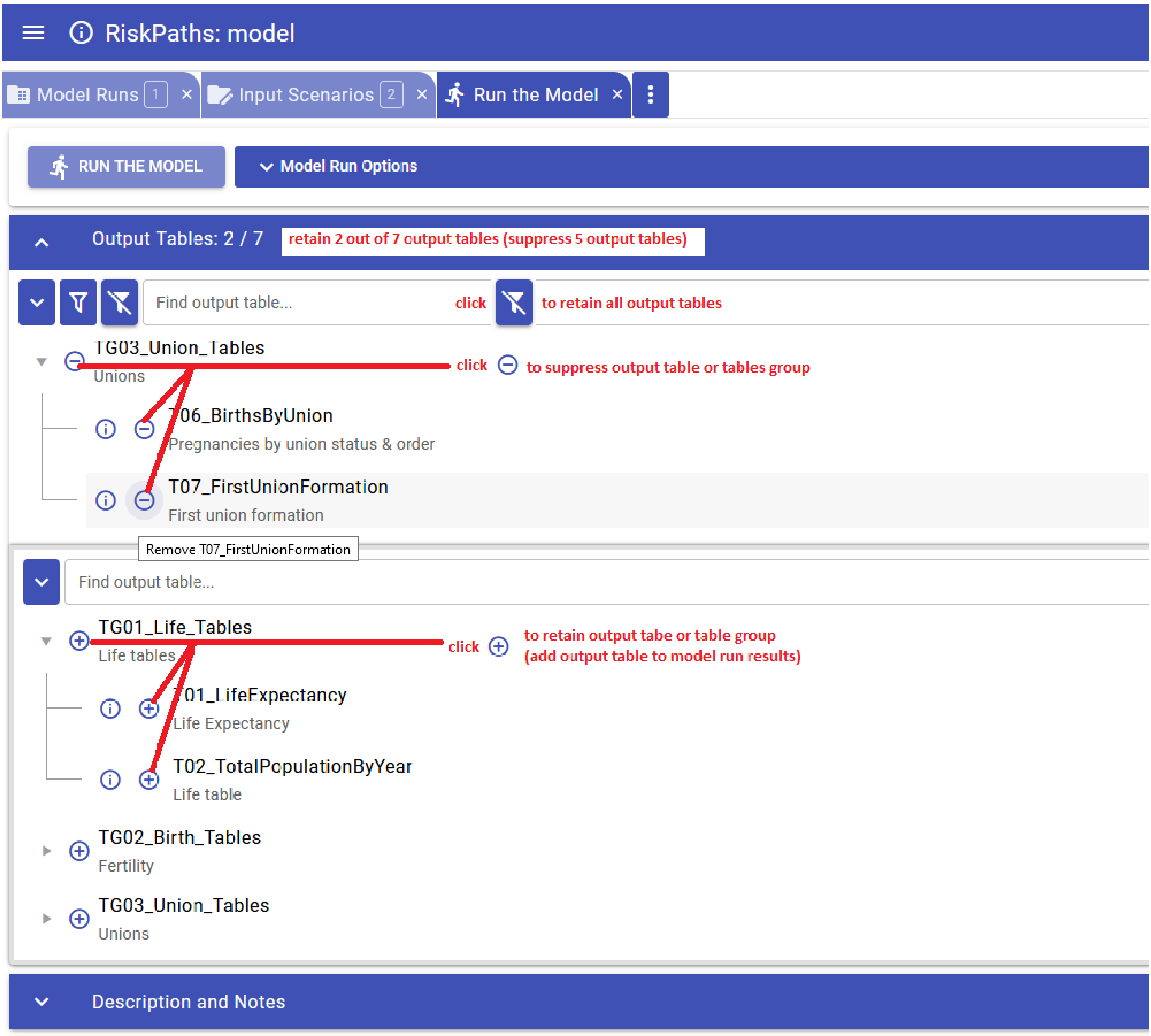
In order to run the the model please click on **Run the Model** tab or select it from the menu and do:

- enter model run name, it must be a valid file name, and cannot contain any of: `` "'`:*?><|${}@&^;/\ ``
- you cannot change run name later, there is no "rename" model run option;
- enter number of sub-values (sub-samples) for your model run;
- **make sure** you have your input scenario check box selected;
- typically your scenario does not contain all model parameters, **make sure** proper base run is selected;
- enter run description and (optional) run notes, you can always edit it later;
- select output tables which you want to retain in your model run results.

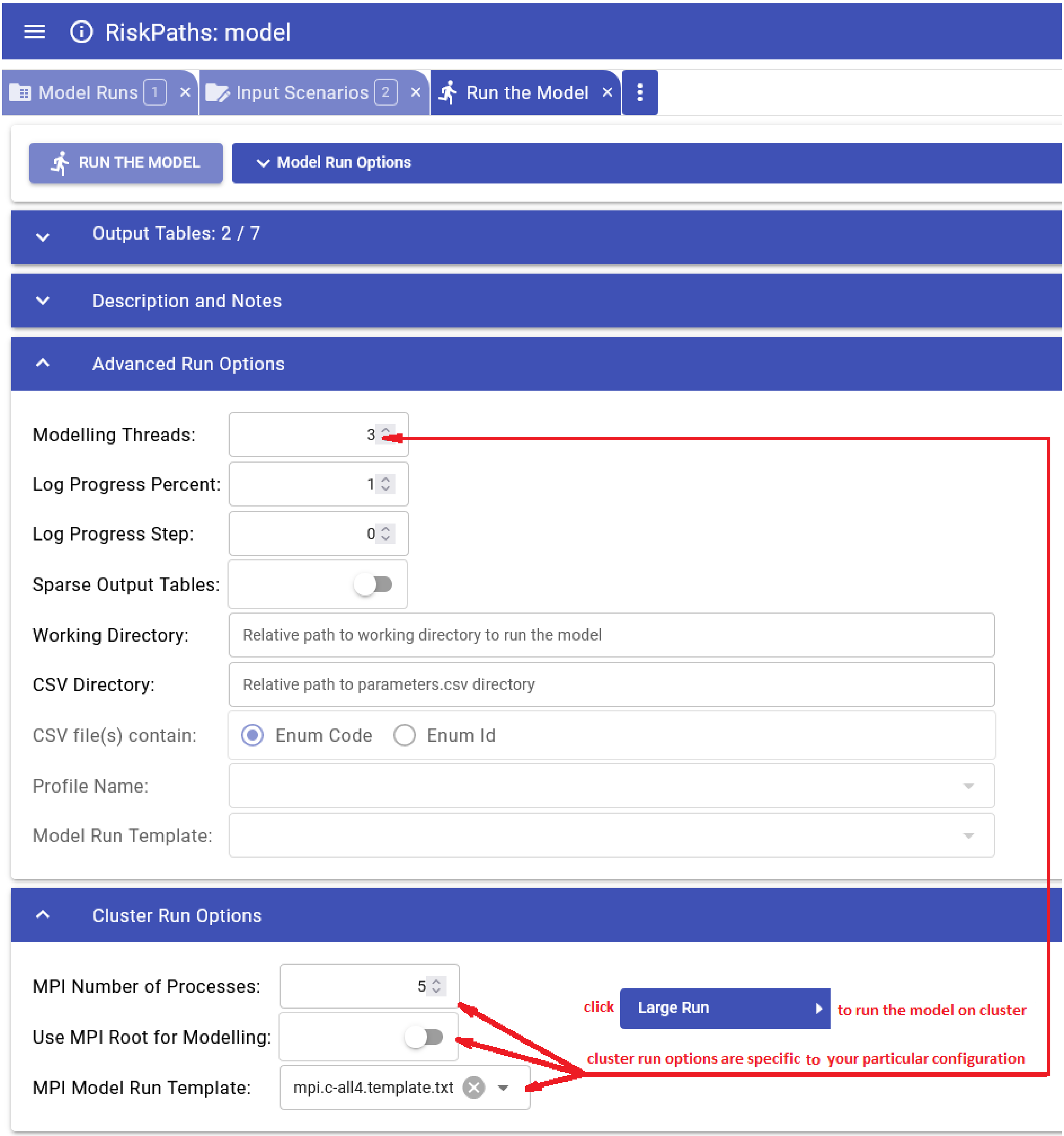
Other (advanced) model run options can be pre-selected by clicking on suitable option button. For example, if you have back-end computational cluster then you may see "Large Run" button, clicking on it sets advanced Cluster Run Options.



Model run output tables selection:



Example of advanced options to run the model on back-end computational cluster:

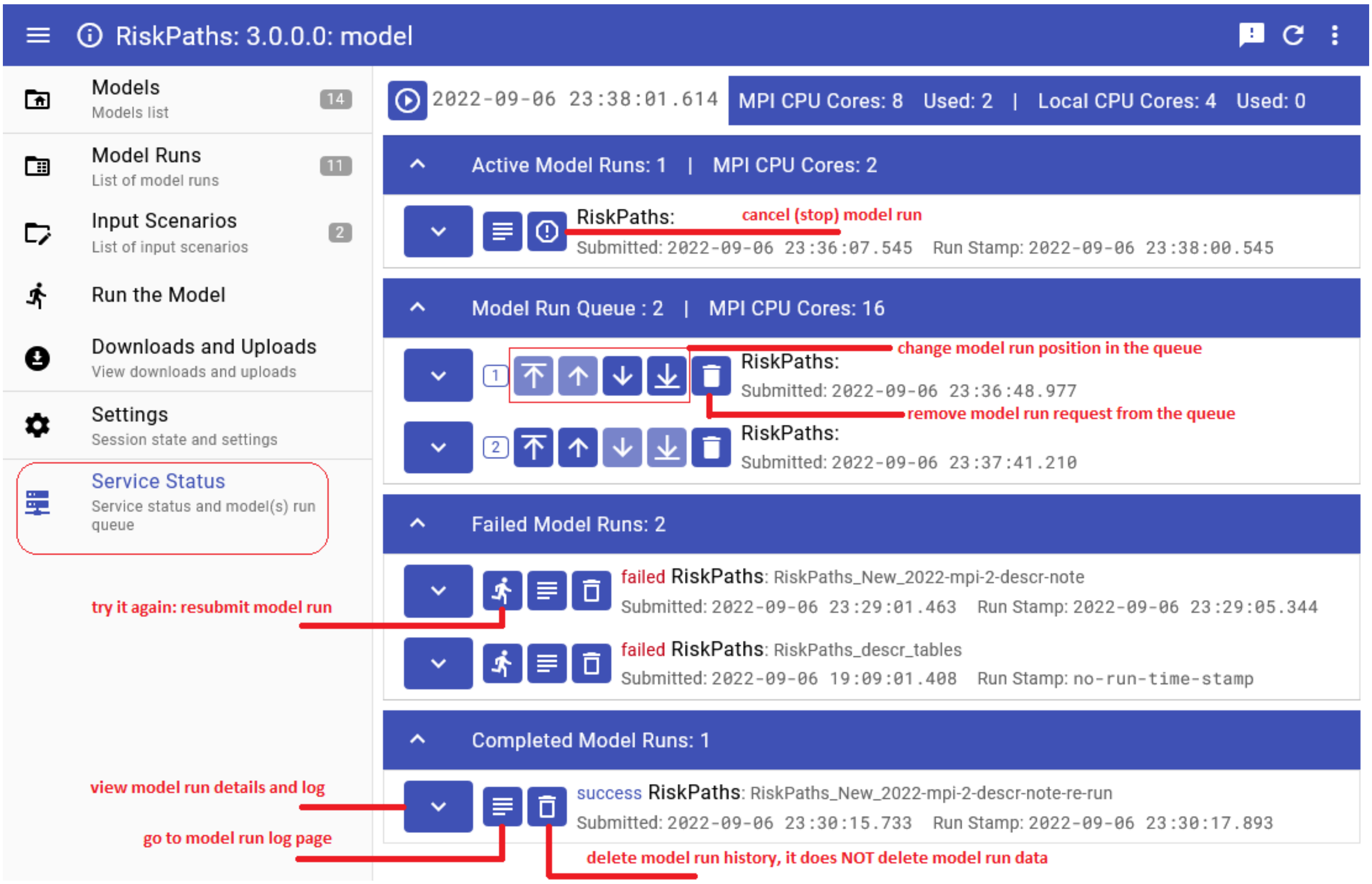


Model run jobs: queue and status:

If model run jobs enabled on your local workstation or in cloud then from Service Status page you can:

- see current model runs queue
- re-arrange your model run queue
- delete your model run job from the queue
- see the history of model runs
- re-submit model run again if it was failed
- see currently active model runs
- cancel (stop) model run

- see all servers status and load



UI Localization: Translation of openM++

Quick Start

To provide translated messages for openM++ UI you should:

- create translated messages file for your language, for example Deutsch: `ompp-ui/src/i18n/de/index.js`
- modify openM++ UI main page `ompp-ui/src/layouts/MainLayout.vue` to support new language
- rebuild openM++ by running `npm run dev` as described at Quick Start for OpenM++ Developers: Build ompp-ui

Please contact us at GitHub openM++ UI project or by email: `openmpp dot org at gmail dot com` for assistance. We certainly can do all necessary steps to include your translation into openM++ UI.

Example of translated messages file

Short fragment from translated messages file `ompp-ui/src/i18n/fr/index.js` for Français language:

```
export default {
  'About': 'À propos',
  'Advanced Run Options': "Options d'exécution avancées",
  'Yes': 'Oui',
  'You have {count} unsaved parameter(s)': 'Vous avez {count} paramètre(s) non enregistré(s)'
}
```

We would appreciate any help with French translation, since person who did it is not a locuteur natif français. Thank you in advance.

OpenM++ UI localization based on internationalization plugin for Vue.js and you can find detailed documentation at that project GitHub page.

How to modify UI main page to include to support new language

Open `ompp-ui/src/layouts/MainLayout.vue` in any text editor and modify following part of the code:

```
  import(
    /* webpackInclude: /(fr|en-us)\.js$/ */
```

to include new language, for example Deutsch:

```
  import(
    /* webpackInclude: /(de|fr|en-us)\.js$/ */
```

Oms: openM++ web-service

What is openM++ web-service

OpenM++ web-service (oms) is a JSON web-service written in Go and used from openM++ UI JavaScript. Today most of popular development platforms (.NET, Java, Python, Perl, R, JavaScript, etc.) with only few lines of code allow to create HTTP client and send-receive JSON data. That makes integration with openM++ very easy.

How to start openM++ web-service

OpenM++ web-service does not required any installation. It can be run with default settings from command-line prompt.

To start openM++ web-service on Windows:

- download and unzip openM++ https://github.com/openmpp/main/releases/latest binaries into `C:\SomeDir\`
- run oms from command-line:

```
C:
cd \SomeDir\openmpp_win_20190508\
bin\oms.exe
```

```
2022-09-14 15:51:30.477 Models directory:    models\bin
2022-09-14 15:51:30.565 HTML UI directory:   html
2022-09-14 15:51:30.567 Etc directory:       etc
2022-09-14 15:51:30.567 Oms instance name:   localhost_4040
2022-09-14 15:51:30.574 Listen at localhost:4040
2022-09-14 15:51:30.574 To start open in your browser: http://localhost:4040
2022-09-14 15:51:30.574 To finish press Ctrl+C
```

OpenM++ UI is a client of `oms` web-service, after above command you can open UI in browser at http://localhost:4040

To start openM++ web-service on Linux:

- download and unpack openM++, i.e.:

```
wget https://github.com/openmpp/main/releases/download/v1.2.0/openmpp_debian_20190508.tar.gz
tar xzf openmpp_debian_20190508.tar.gz
```

- run oms executable:

```
cd openmpp_debian_20190508/
bin/oms
```

```
2022-09-14 15:51:30.477 Models directory:    models/bin
2022-09-14 15:51:30.565 HTML UI directory:   html
2022-09-14 15:51:30.567 Etc directory:       etc
2022-09-14 15:51:30.567 Oms instance name:   localhost_4040
2022-09-14 15:51:30.574 Listen at localhost:4040
2022-09-14 15:51:30.574 To start open in your browser: http://localhost:4040
2022-09-14 15:51:30.574 To finish press Ctrl+C
```

Note: We recommend to use normal Windows command line cmd.exe. If you are using Windows PowerShell then it may be necessary to put "quotes" around command line options, e.g:

```
oms.exe "-oms.ApiOnly"
```

Oms as "pure" web-service vs "full" web-UI

By default `oms.exe` started in "full" web-UI mode. That means it handles web-service requests and web-UI content from `./html` sub-directory. If you want only "pure" web-service mode without UI then use:

```
oms -oms.ApiOnly
```

How to use oms: arguments of web-service methods

Following arguments most often used in web-service methods:

:model - model digest or model name

Example of method:

```
GET /api/model/:model
```

Call example:

```
http://localhost:4040/api/model/f5024ac32c4e8abfc696a0f925141c95
http://localhost:4040/api/model/modelOne
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

:run - model run or model task run

Example of method:

```
GET /api/model/:model/run/:run/status
GET /api/model/:model/task/:task/run-status/run/:run
```

Call example:

```
http://localhost:4040/api/model/modelOne/run/modelOne_first_run/status
http://localhost:4040/api/model/modelOne/run/d06f4a0a45a9514c22593025e489f933/status
http://localhost:4040/api/model/modelOne/task/taskOne/run-status/run/First Task Run
```

This argument is used to identify model run or modeling task run.

Modeling task run can be identified by task run stamp or task run name.

Model run can be identified by run digest, run stamp or run name. It is recommended to use run digest because it is uniquely identifies model run. Run stamp can be explicitly specified as command line option when you run the model. If run stamp not specified then it is automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use run name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

:lang - language code

Example of method:

```
GET /api/model/:model/text/lang/:lang
```

Call example:

```
http://localhost:4040/api/model/modelOne/text/lang/EN
http://localhost:4040/api/model/modelOne/text/lang/en_US
```

Language code can be a model language (ex.: EN, FR) or any MIME language (see BCP47 or RFC3282). If no language explicitly specified then `Accept-Language` header is used (supplied by browser).

Result returned in best matched language supported by model. For example for en_US result is model language EN, if model supported EN language. If no such language then result is in default model language or can be empty.

:set - set of input data (a.k.a. workset)

Method examples:

```
GET  /api/model/:model/workset/:set/status
POST /api/model/:model/workset/:set/readonly/:val
```

Call examples:

```
http://localhost:4040/api/model/modelOne/workset/modelOne_set/status
curl -v -X POST http://localhost:4040/api/model/modelOne/workset/modelOne_set/readonly/1
```

Workset is a set of model input parameters (a.k.a. "scenario" input) and it used to run the model Each model workset uniquely identified by name.

:task - modelling task

Method examples:

```
GET /api/model/:model/task/:task/text/lang=FR
```

Call examples:

```
http://localhost:4040/api/model/modelOne/task/taskOne/text
curl -v http://localhost:4040/api/model/modelOne/task/taskOne/text/lang=fr_CA
```

Modelling task consists of multiple input data sets (a.k.a. worksets or scenarios in Modgen). Task can be used to run the model in batch mode.

:profile - set of key-value options

Method examples:

```
GET /api/model/:model/profile/:profile
POST /api/model/:model/profile/:profile/key/:key/value/:value
```

Call examples:

```
http://localhost:4040/api/model/modelOne/profile/modelOne
curl -v -X POST http://localhost:4040/api/model/modelOne/profile/m1/key/Parameter.StartingSeed/value/4095
```

Profile is a set of key-value options and it used to run the model. Each profile uniquely identified by profile name. Each profile can include multiple key-value options.

Results of web-service methods

Run status

Model run status and task run status may contain one of the following values:

```
i = initial state, not running yet
p = run in progress
w = task wait for additional input
s = completed successfully
x = completed by exit (reserved fro internal use)
e = completed with error
```

Important: if model run failed with exception (e.g. database write exception) then status may not be updated and still `p=in progress`.

Oms web-service configuration

Oms default configuration options can be overwritten by command-line arguments or ini-file. For example:

- listen from any host on port 7070:

  ```
  oms -l :7070
  ```

- serve only API calls and not `html` for openM++ UI:

  ```
  oms -oms.ApiOnly
  ```

- listen from localhost port 4044 only and read more oms run options from `oms.ini` file:

  ```
  oms -l localhost:4044 -ini oms.ini
  ```

- models directory relative path is: `../some/dir`

  ```
  oms -oms.ModelDir ../some/dir
  ```

- typical log settings for remote server:
  - log user request
  - log into the file instead of cosole by default
  - log files rotation: create new log file every day

    ```
    oms -l localhost:4044 -oms.LogRequest -OpenM.LogToConsole false -OpenM.LogToFile -OpenM.LogUseDailyStamp
    ```

- typical settings for model user in cloud:
  - allow user home directory with downloads and uploads
  - use model run jobs to manage back-end computational servers resources

    ```
    oms -l localhost:4044 -oms.HomeDir models/home -oms.AllowDownload -oms.AllowUpload -oms.JobDir job
    ```

    It is recommended to use `oms.ini` file to avoid long command lines, especially for cloud environment where you may want to combine log options and user options from two examples above.

Get and use oms web-service configuration

Clients of oms web-service can retrieve configuration by calling GET web-service configuration or simply by open http://localhost:4040/api/service/config in the browser. Response to that call may also contain client environment variables which names started from `OM_CFG_` prefix *(`oms` web-service does not use any of `OM_CFG_` environment variables, it only passes it to clients)*.

For example openM++ UI uses following server variables:

```
OM_CFG_LOGIN_URL=/public/login_required.html
OM_CFG_LOGOUT_URL=/login?logout=true
OM_CFG_DEFAULT_RUN_TMPL=run.Win32.Debug.template.txt
OM_CFG_INI_ALLOW=true
OM_CFG_INI_ANY_KEY=true
```

OpenM++ UI is using above variables as follow:

- `OM_CFG_LOGIN_URL` : display user login button linked to the URL
- `OM_CFG_LOGOUT_URL` : display user logout button linked to the URL
- `OM_CFG_DEFAULT_RUN_TMPL` : use this template to run the model, e.g.: to debug from IDE
- `OM_CFG_INI_ALLOW` : allow user to run the model with ini-file, e.g.: `RiskPaths.ini`
- `OM_CFG_INI_ANY_KEY` : allow to use model development options from ini-file

Note: Model ini-files and model development options described at: Model Run Options and ini file.

Oms run options

Following options supported by oms:

```
-oms.Listen:        address to listen, default: localhost:4040
-l:                 address to listen (short form of -oms.Listen)
-OpenM.IniFile:     path to ini-file
-ini ini-file:      path to ini-file (short of OpenM.IniFile)
-oms.ApiOnly:       if true then API only web-service, no web UI
-oms.RootDir:       oms root directory, default: current directory
-oms.ModelDir:      models directory, if relative then must be relative to root directory, default: models/bin
-oms.ModelLogDir:   models log directory, if relative then must be relative to root directory: default: "models/log
-oms.HomeDir:       user personal home directory, if relative then must be relative to root directory
-oms.AllowDownload: if true then allow download from user home/io/download directory
-oms.AllowUpload:   if true then allow upload to user home/io/upload directory
-oms.HtmlDi:        front-end UI directory, if relative then must be relative to root directory, default: html
-oms.EtcDir:        configuration files directory, if relative then must be relative to oms root directory, default: etc
-oms.JobDir:        model run jobs directory, if relative then must be relative to root directory
-oms.Name:          oms instance name, used model run by jobs, automatically generated if empty
-oms.UrlSaveTo:     file path to save oms URL in form of: http://localhost:4040, if relative then must be relative to root directory
-oms.Languages:     comma-separated list of supported languages, default: en
-oms.CodePage:      code page to convert source file into utf-8, e.g.: windows-1252
-oms.DoubleFormat:  format to convert float or double value to string, default: %.15g
-oms.MaxRowCount:   max number of rows to return from read parameters or output tables, default: 100
-oms.MaxRunHistory: max number of model runs to keep in run list history, default: 1000

-OpenM.LogToConsole:     if true then log to standard output (default true)
-v:                      if true then log to standard output (short of OpenM.LogToConsole)
-OpenM.LogToFile:        if true then log to file
-OpenM.LogFilePath:      path to log file, default = current/dir/oms.log
-OpenM.LogUseDailyStamp: if true then use daily-stamp in log file name
-OpenM.LogUsePidStamp:   if true then use process id stamp in log file name
-OpenM.LogUseTimeStamp:  if true then use time-stamp in log file name
-OpenM.LogSql:           if true then log sql statements into log file
-oms.LogRequest:         if true then log HTTP requests
```

There are many common options, e.g.: `-OpenM.LogToFile` which can be used with any openM++ executable: models, compiler, dbcopy and oms.

Example of oms.ini

```
; This is a comment, started with ; semicolon
# This is a comment, started with # hash
; boolean options can be "true" or "false"

[oms]
Listen        = localhost:4040 # address to listen, default: localhost:4040
RootDir       =                # oms "root" directory, expected to have log subfolder
ModelDir      = models/bin     # models executable and model.sqlite directory, if relative then must be relative to oms root directory
ModelLogDir   = models/log     # models log directory, if relative then must be relative to oms root directory
HomeDir       = models/home    # user personal home directory, if relative then must be relative to oms root directory
AllowDownload = false          # if true then allow download from user home sub-directory: home/io/download
AllowUpload   = false          # if true then allow upload to user home sub-directory: home/io/upload
ApiOnly       = false          # if true then API only web-service, no web UI
HtmlDir       = html           # front-end web UI directory, if relative then must be relative to oms root directory
EtcDir        = etc            # configuration files directory, if relative then must be relative to oms root directory
JobDir        =                # jobs control directory, if empty then jobs control disabled
Name          =                # instance name, used for job control
UrlSaveTo     =                # file path to save oms URL, if relative then must be relative to oms root directory
LogRequest    = false          # if true then log HTTP requests
Languages     = en             # comma-separated list of supported languages
CodePage      =                # code page to convert source file into utf-8, e.g.: windows-1252
MaxRowCount   = 100            # max number of rows to return from read parameters or output tables
MaxRunHistory = 1000           # max number of model runs to keep in run list history
DoubleFormat  = %.15g          # format to convert float or double value to string, e.g. %.15g

; Log settings:
; log can be enabled/disabled for 3 independent streams:
;   console             - standard output
;   "current" log file  - log file with specified name, overwritten on every model run
;   "stamped" log file  - log file with unique name, created for every model run
;
; "stamped" name produced from "current" name by adding time-stamp and/or process-id-stamp, i.e.:
;   oms.log => oms.2012_08_17_16_04_59_148.123456.log
#
; LogUseDailyStamp creates new log file every day
; by default LogUseDailyStamp:
;   = false if log file disabled (default)
;   = false if "stamped" log file enabled
;   = true  if log file enabled and "stamped" log file disabled
;
[OpenM]
LogToConsole = true      # if true then log to standard output
LogToFile = false        # if true then log to file
LogFilePath = oms.log    # log file path, default = current/dir/exeName.log
LogUseTimeStamp = false  # if true then use time-stamp in log file name
LogUsePidStamp = false   # if true then use pid-stamp in log file name
LogUseDailyStamp = false # if true then use daily-stamp in log file name
LogSql = false           # if true then log sql statements into log file
```

Oms directory structure: user home and jobs directories

Following directory structure expected by default:

```
./        -> oms "root" directory, by default it is current directory
    html/    -> web-UI directory with HTML, js, css, images...
    etc/     -> config files directory, contain template(s) to run models
    log/     -> recommended log files directory
    models/
          bin/  -> default model.exe and model.sqlite directory
          log/  -> default directory for models run log files
```

If you don't don't want web-UI or don't have `html` directory then start oms as:

```
oms -oms.ApiOnly
```

You can explicitly specify oms log files location, models and models log directory, e.g.:

```
oms -oms.ModelDir /my-models -oms.ModelLogDir /my-models-log
```

If you want to use log file and no console messages:

```
oms -OpenM.LogToConsole=false -OpenM.LogToFile
oms -OpenM.LogToConsole=false -OpenM.LogFilePath log/oms.log
```

If you want to use "daily" log files:

```
oms -OpenM.LogUseDailyStamp -OpenM.LogToFile
oms -OpenM.LogUseDailyStamp -OpenM.LogFilePath log/oms.log
```

User home directory

You can enable user home directory to store home directory for user personal settings, downloads of model model run results or upload input scenarios:

```
oms -oms.HomeDir models/home -oms.AllowDownload -oms.AllowUpload
```

Above command assume directory structure with `home`, `download` and `upload` sub-folders of `models`:

```
./        -> oms "root" directory, by default it is current directory
    html/    -> web-UI directory with HTML, js, css, images...
    etc/     -> config files directory, contain template(s) to run models
    log/     -> recommended log files directory
    models/
          bin/  -> default model.exe and model.sqlite directory
          log/  -> default directory for models run log files
          home/ -> user personal home directory
              io/download  -> user directory for download files
            io/upload    -> user directory to upload files
```

Note: openM++ `dbcopy` utility is required for download and upload, it must be located in the same directory where `oms` executable is.

Model run jobs directory structure

If you want to have model runs queue, or using openM++ in cloud and want automatically scale up and down cloud resources, e.g. start and stop virtual machines for model runs then start `oms` with job control option:

```
oms -oms.JobDir job
```

Following directory structure expected:

```
./        -> oms "root" directory, by default it is current directory
    html/    -> web-UI directory with HTML, js, css, images...
    etc/     -> config files directory, contain template(s) to run models
    log/     -> recommended log files directory
    models/
          bin/  -> default model.exe and model.sqlite directory
          log/  -> default directory for models run log files
            home/ -> user personal home directory
                io/download  -> user directory for download files
              io/upload    -> user directory to upload files
    job/  -> model run jobs control directory
          job.ini   -> (optional) job control settings
          active/   -> active model run state files
          history/  -> model run history files
          queue/    -> model run queue files
          state/    -> jobs state and computational servers state files
               jobs.queue.paused  -> if such file exists then jobs queue is paused
```

Please visit following page to find out how to use oms in cloud and manage model runs queue.

Oms: openM++ web-service API

Web-service methods arguments

```
:model   - model digest or model name
:lang    - language code
:run     - model run digest, run stamp or run name, modeling task run stamp or task run name
:set     - name of workset (input set of model parameters)
:profile - profile name
:task    - modeling task
```

See more details at: Arguments of web-service methods.

GET Model Metadata

GET model list

```
GET /api/model-list
```

GET model list including text (description and notes)

```
GET /api/model-list/text
GET /api/model-list/text/lang/:lang
```

GET model definition metadata

```
GET /api/model/:model
```

GET model metadata including text (description and notes)

```
GET /api/model/:model/text
GET /api/model/:model/text/lang/:lang
```

GET model metadata including text in all languages

```
GET /api/model/:model/text/all
```

GET Model Extras

GET model languages

```
GET /api/model/:model/lang-list
```

GET model language-specific strings

```
GET /api/model/:model/word-list
GET /api/model/:model/word-list/lang/:lang
```

GET model profile

```
GET /api/model/:model/profile/:profile
```

GET list of profiles

```
GET /api/model/:model/profile-list
```

GET Model Run results metadata

GET list of model runs

```
GET /api/model/:model/run-list
```

GET list of model runs including text (description and notes)

```
GET /api/model/:model/run-list/text
GET /api/model/:model/run-list/text/lang/:lang
```

GET status of model run

```
GET /api/model/:model/run/:run/status
```

GET status of model run list

```
GET /api/model/:model/run/:run/status/list
```

GET status of first model run

```
GET /api/model/:model/run/status/first
```

GET status of last model run

```
GET /api/model/:model/run/status/last
```

GET status of last completed model run

```
GET /api/model/:model/run/status/last-completed
```

GET model run metadata and status

```
GET /api/model/:model/run/:run
```

GET model run including text (description and notes)

```
GET /api/model/:model/run/:run/text
GET /api/model/:model/run/:run/text/lang/:lang
```

GET model run including text in all languages

```
GET /api/model/:model/run/:run/text/all
```

GET Model Workset metadata: set of input parameters

GET list of model worksets

```
GET /api/model/:model/workset-list
```

GET list of model worksets including text (description and notes)

```
GET /api/model/:model/workset-list/text
GET /api/model/:model/workset-list/text/lang/:lang
```

GET workset status

```
GET /api/model/:model/workset/:set/status
GET /api/model/:model/workset/:set
```

GET model default workset status

```
GET /api/model/:model/workset/status/default
```

GET workset including text (description and notes)

```
GET /api/model/:model/workset/:set/text
GET /api/model/:model/workset/:set/text/lang/:lang
```

GET workset including text in all languages

```
GET /api/model/:model/workset/:set/text/all
```

Read Parameters, Output Tables or Microdata values

Read parameter values from workset

```
POST /api/model/:model/workset/:set/parameter/value
```

Read parameter values from workset (enum id's)

```
POST /api/model/:model/workset/:set/parameter/value-id
```

Read parameter values from model run

```
POST /api/model/:model/run/:run/parameter/value
```

Read parameter values from model run (enum id's)

```
POST /api/model/:model/run/:run/parameter/value-id
```

Read output table values from model run

```
POST /api/model/:model/run/:run/table/value
```

Read output table values from model run (enum id's)

```
POST /api/model/:model/run/:run/table/value-id
```

Read microdata values from model run

```
POST /api/model/:model/run/:run/microdata/value
```

Read microdata values from model run (enum id's)

```
POST /api/model/:model/run/:run/microdata/value-id
```

GET Parameters, Output Tables or Microdata values

GET parameter values from workset

```
GET /api/model/:model/workset/:set/parameter/:name/value
GET /api/model/:model/workset/:set/parameter/:name/value/start/:start
GET /api/model/:model/workset/:set/parameter/:name/value/start/:start/count/:count
```

GET parameter values from model run

```
GET /api/model/:model/run/:run/parameter/:name/value
GET /api/model/:model/run/:run/parameter/:name/value/start/:start
GET /api/model/:model/run/:run/parameter/:name/value/start/:start/count/:count
```

GET output table expression(s) from model run

```
GET /api/model/:model/run/:run/table/:name/expr
GET /api/model/:model/run/:run/table/:name/expr/start/:start
GET /api/model/:model/run/:run/table/:name/expr/start/:start/count/:count
```

GET output table accumulator(s) from model run

```
GET /api/model/:model/run/:run/table/:name/acc
GET /api/model/:model/run/:run/table/:name/acc/start/:start
GET /api/model/:model/run/:run/table/:name/acc/start/:start/count/:count
```

GET output table all accumulators from model run

```
GET /api/model/:model/run/:run/table/:name/all-acc
GET /api/model/:model/run/:run/table/:name/all-acc/start/:start
GET /api/model/:model/run/:run/table/:name/all-acc/start/:start/count/:count
```

GET microdata values from model run

```
GET /api/model/:model/run/:run/microdata/:name/value
GET /api/model/:model/run/:run/microdata/:name/value/start/:start
GET /api/model/:model/run/:run/microdata/:name/value/start/:start/count/:count
```

GET Parameters, Output Tables or Microdata values as CSV

GET csv parameter values from workset

```
GET /api/model/:model/workset/:set/parameter/:name/csv
GET /api/model/:model/workset/:set/parameter/:name/csv-bom
```

GET csv parameter values from workset (enum id's)

```
GET /api/model/:model/workset/:set/parameter/:name/csv-id
GET /api/model/:model/workset/:set/parameter/:name/csv-id-bom
```

GET csv parameter values from model run

```
GET /api/model/:model/run/:run/parameter/:name/csv
GET /api/model/:model/run/:run/parameter/:name/csv-bom
```

GET csv parameter values from model run (enum id's)

```
GET /api/model/:model/run/:run/parameter/:name/csv-id
GET /api/model/:model/run/:run/parameter/:name/csv-id-bom
```

GET csv output table expressions from model run

```
GET /api/model/:model/run/:run/table/:name/expr/csv
GET /api/model/:model/run/:run/table/:name/expr/csv-bom
```

GET csv output table expressions from model run (enum id's)

```
GET /api/model/:model/run/:run/table/:name/expr/csv-id
GET /api/model/:model/run/:run/table/:name/expr/csv-id-bom
```

GET csv output table accumulators from model run

```
GET /api/model/:model/run/:run/table/:name/acc/csv
GET /api/model/:model/run/:run/table/:name/acc/csv-bom
```

GET csv output table accumulators from model run (enum id's)

```
GET /api/model/:model/run/:run/table/:name/acc/csv-id
GET /api/model/:model/run/:run/table/:name/acc/csv-id-bom
```

GET csv output table all accumulators from model run

```
GET /api/model/:model/run/:run/table/:name/all-acc/csv
GET /api/model/:model/run/:run/table/:name/all-acc/csv-bom
```

GET csv output table all accumulators from model run (enum id's)

```
GET /api/model/:model/run/:run/table/:name/all-acc/csv-id
GET /api/model/:model/run/:run/table/:name/all-acc/csv-id-bom
```

GET csv microdata values from model run

```
GET /api/model/:model/run/:run/microdata/:name/csv
GET /api/model/:model/run/:run/microdata/:name/csv-bom
```

GET csv microdata values from model run (enum id's)

```
GET /api/model/:model/run/:run/microdata/:name/csv-id
GET /api/model/:model/run/:run/microdata/:name/csv-id-bom
```

GET Modeling Task metadata and task run history

GET list of modeling tasks

```
GET /api/model/:model/task-list
```

GET list of modeling tasks including text (description and notes)

```
GET /api/model/:model/task-list/text
GET /api/model/:model/task-list/text/lang/:lang
```

GET modeling task input worksets

```
GET /api/model/:model/task/:task/sets
```

GET modeling task run history

```
GET /api/model/:model/task/:task/runs
```

GET status of modeling task run

```
GET /api/model/:model/task/:task/run-status/run/:run
```

GET status of modeling task run list

```
GET /api/model/:model/task/:task/run-status/list/:run
```

GET status of modeling task first run

```
GET /api/model/:model/task/:task/run-status/first
```

GET status of modeling task last run

```
GET /api/model/:model/task/:task/run-status/last
```

GET status of modeling task last completed run

```
GET /api/model/:model/task/:task/run-status/last-completed
```

GET modeling task including text (description and notes)

```
GET /api/model/:model/task/:task/text
GET /api/model/:model/task/:task/text/lang/:lang
```

GET modeling task text in all languages

```
GET /api/model/:model/task/:task/text/all
```

Update Model Profile: set of key-value options

PATCH create or replace profile

```
PATCH /api/model/:model/profile
```

DELETE profile

```
DELETE /api/model/:model/profile/:profile
```

POST create or replace profile option

```
POST /api/model/:model/profile/:profile/key/:key/value/:value
```

DELETE profile option

```
DELETE /api/model/:model/profile/:profile/key/:key
```

Update Model Workset: set of input parameters

POST update workset read-only status

```
POST /api/model/:model/workset/:set/readonly/:readonly
```

PUT create new workset

```
PUT  /api/workset-create
```

PUT create or replace workset

```
PUT /api/workset-replace
```

PATCH create or merge workset

```
PATCH /api/workset-merge
```

DELETE workset

```
DELETE /api/model/:model/workset/:set
```

DELETE parameter from workset

```
DELETE /api/model/:model/workset/:set/parameter/:name
```

PATCH update workset parameter values

```
PATCH /api/model/:model/workset/:set/parameter/:name/new/value
```

PATCH update workset parameter values (enum id's)

```
PATCH /api/model/:model/workset/:set/parameter/:name/new/value-id
```

PATCH update workset parameter(s) value notes

```
PATCH /api/model/:model/workset/:set/parameter-text
```

PUT copy parameter from model run into workset

```
PUT /api/model/:model/workset/:set/copy/parameter/:name/from-run/:run
```

PATCH merge parameter from model run into workset

```
PATCH /api/model/:model/workset/:set/merge/parameter/:name/from-run/:run
```

PUT copy parameter from workset to another

```
PUT /api/model/:model/workset/:set/copy/parameter/:name/from-workset/:from-set
```

PATCH merge parameter from workset to another

```
PATCH /api/model/:model/workset/:set/merge/parameter/:name/from-workset/:from-set
```

Update Model Runs

PATCH update model run text (description and notes)

```
PATCH /api/run/text
```

DELETE model run

```
DELETE /api/model/:model/run/:run
```

PATCH update run parameter(s) value notes

```
PATCH /api/model/:model/run/:run/parameter-text
```

Update Modeling Tasks

PUT create or replace modeling task

```
PUT  /api/task-new
```

PATCH create or update modeling task

```
PATCH /api/task
```

DELETE modeling task

```
DELETE /api/model/:model/task/:task
```

Run Models: run models and monitor progress

POST a request to run the model

```
POST /api/run
```

GET state of current model run

```
GET /api/run/log/model/:model/stamp/:stamp
GET /api/run/log/model/:model/stamp/:stamp/start/:start/count/:count
```

PUT stop model run

```
PUT /api/run/stop/model/:model/stamp/:stamp
```

Download model, model run results or input parameters

GET download log file

```
GET /api/download/log/file/:name
```

GET all download log files for the model

```
GET /api/download/log/model/:model
```

GET all download log files

```
GET /api/download/log/all
```

GET download files tree

```
GET /api/download/file-tree/:folder
```

POST initiate model download

```
POST /api/download/model/:model
```

POST initiate model run download

```
POST /api/download/model/:model/run/:run
```

POST initiate model workset download

```
POST /api/download/model/:model/workset/:set
```

DELETE download files

```
DELETE /api/download/delete/:folder
```

DELETE download files asynchronously

```
DELETE /api/download/start/delete/:folder
```

Upload model runs or worksets

GET upload log file

```
GET /api/upload/log/file/:name
```

GET all upload log files for the model

```
GET /api/upload/log/model/:model
```

GET all upload log files

```
GET /api/upload/log/all
```

GET upload files tree

```
GET /api/upload/file-tree/:folder
```

POST initiate model run upload

```
POST /api/upload/model/:model/run
POST /api/upload/model/:model/run/:run
```

POST initiate workset upload

```
POST /api/upload/model/:model/workset
POST /api/upload/model/:model/workset/:set
```

DELETE upload files

```
DELETE /api/upload/delete/:folder
```

DELETE upload files asynchronously

```
DELETE /api/upload/start/delete/:folder
```

User: manage user settings and data

GET user views for the model

```
GET /api/user/view/model/:model
```

PUT user views for the model

```
PUT /api/user/view/model/:model
```

DELETE user views for the model

```
DELETE /api/user/view/model/:model
```

Model run jobs and service state

GET web-service configuration

```
GET /api/service/config
```

GET web-service state

```
GET /api/service/state
```

GET state of active model run job

```
GET /api/service/job/active/:job
```

GET state of model run job from queue

```
GET /api/service/job/queue/:job
```

GET state of model run job from history

```
GET /api/service/job/history/:job
```

PUT model run job into other queue position

```
PUT /api/service/job/move/:pos/:job
```

DELETE state of model run job from history

```
DELETE /api/service/job/delete/history/:job
```

Administrative: manage web-service state

POST a request to refresh models catalog

```
POST /api/admin/all-models/refresh
```

POST a request to close models catalog

```
POST /api/admin/all-models/close
```

POST a request to pause model run queue

```
POST /api/admin/jobs-pause/:pause
```

PUT a request to shutdown web-service

```
PUT /api/admin/shutdown
```

Oms: How to prepare model input parameters

Overview

OpenM++ provides multiple different ways to supply input parameters and run the models as described at:

- Model Run Cycle: How model finds input parameters
- Model Run: How to Run the Model

You don't have to do any programming or database operations in order to provide model input parameters, you can:

- provide parameter value as command line argument
- run model with default workset (default "scenario")
- use workset name ("scenario" name) to run the model
- use ini-file to provide model parameters
- supply parameter values as csv-file(s)

Also following API available for advanced parameter manipulation and output results processing:

- JSON web-service to use with any modern framework (.NET, JavaScript, Python, etc.)
- Go library and tools
- OpenMpp R package and R usage examples

Current page describe an usage of openM++ JSON web-service (oms) in order to prepare, examine and modify model input parameters. There are two terms are used in text below: "workset" and "base run". Please see Model Run Cycle: How model finds input parameters page for details.

Workset: set of model input parameters (a.k.a. "scenario")

Workset is a set of model input parameters in database which we can use to run the model. Each workset has unique name. Each model must have "default workset", which is a first set of model input parameters. Model user can create, modify and delete worksets.

Base run

Each model run started from creating full copy of model input parameters in database, which are used for that particular model run. Because usually only small portion of model parameters are changing between model runs it is convenient to create new workset (new "scenario") based on input parameters of previous model run, which is called "base run". In that case we only need to supply few modified parameter values and the rest is coming from "base run" parameters.

Start Oms: OpenM++ JSON web-service

Below we are using oms web-service to prepare model input. Most examples are created with browser output and `curl` to avoid any unnecessary programing languages details.

You can start oms web-service on Windows:

```
C:
cd \SomeDir\openmpp_win_20190508\
bin\oms.exe -oms.ApiOnly
```

Or Linux:

```
cd openmpp_debian_20190508/
bin/oms -oms.ApiOnly
```

If your models are not in `models/bin` sub-folder then use:

```
bin/oms -oms.ApiOnly -oms.ModelDir ../my_model_dir
```

Please see Oms web-service page for more details.

Get list of published models

We need to know model name at least, or better model digest to find or modify model input parameters. Open your favorite browser and type:

```
http://localhost:4040/api/model-list
```

Result can look like:


```
[
  {
    "ModelId": 1,
    "Name": "modelOne",
    "Digest": "_201208171604590148_",
    "Type": 0,
    "Version": "1.0",
    "CreateDateTime": "2012-08-17 16:04:59.0148",
    "DefaultLangCode": "EN"
  },
  {
    "ModelId": 101,
    "Name": "RiskPaths",
    "Digest": "db6e5168c74a73a4f5b194cb2a793444",
    "Type": 0,
    "Version": "3.0.0.0",
    "CreateDateTime": "2018-12-14 18:36:05.0272",
    "DefaultLangCode": "EN"
  }
]
```


Get list of model worksets (set of input parameters, a.k.a. "scenarios")

Go to:

```
http://localhost:4040/api/model/modelOne/workset-list

[
  {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "Default",
    "BaseRunDigest": "",
    "IsReadonly": true,
    "UpdateDateTime": "2013-05-29 23:55:07.1234",
    "Txt": [],
    "Param": []
  },
........
]
```

Model default set of input parameters

First workset is a default set of model input parameters. You can explore more it at

```
http://localhost:4040/api/model/modelOne/workset/Default/text
```

and look at each parameter values, for example:

```
http://localhost:4040/api/model/modelOne/workset/Default/parameter/StartingSeed/value
```


```
[
  {
    "Dims": [],
    "IsNull": false,
    "Value": 8191,
    "SubId": 0
  }
]
```


Or

```
http://localhost:4040/api/model/modelOne/workset/Default/parameter/ageSex/value
```

```
[
{"Dims":["10-20","M"],"IsNull":false,"Value":0.1,"SubId":0}
,{"Dims":["10-20","F"],"IsNull":false,"Value":0.2,"SubId":0}
,{"Dims":["20-30","M"],"IsNull":false,"Value":0.3,"SubId":0}
,{"Dims":["20-30","F"],"IsNull":false,"Value":0.4,"SubId":0}
,{"Dims":["30-40","M"],"IsNull":false,"Value":0.5,"SubId":0}
,{"Dims":["30-40","F"],"IsNull":false,"Value":0.6,"SubId":0}
,{"Dims":["40+","M"],"IsNull":false,"Value":0.7,"SubId":0}
,{"Dims":["40+","F"],"IsNull":false,"Value":0.8,"SubId":0}
]
```


Model run results and run input parameters

To see the history of model runs:

```
http://localhost:4040/api/model/modelOne/run-list
```



```
[
 {
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "Default",
  "SubCount": 1,
  "SubStarted": 1,
  "SubCompleted": 1,
  "CreateDateTime": "2019-01-10 18:36:13.0655",
  "Status": "s",
  "UpdateDateTime": "2019-01-10 18:36:13.0669",
  "Digest": "6fbad822cb9ae42deea1ede626890711",
  "Txt": [],
  "Opts": {},
  "Param": [],
  "Progress": []
 },
........
 {
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "Parameter sub-values 2 from csv",
  "SubCount": 2,
  "SubStarted": 2,
  "SubCompleted": 2,
  "CreateDateTime": "2019-01-10 18:36:13.0745",
  "Status": "s",
  "UpdateDateTime": "2019-01-10 18:36:13.0762",
  "Digest": "ac72e96b549638d31acaf6ee965b23c2",
  "Txt": [],
  "Opts": {},
  "Param": [],
  "Progress": []
 },
........
]
```


Model run can be uniquely identified by run digest, for example above:

- digest: `"ac72e96b549638d31acaf6ee965b23c2"`, run name: "Parameter sub-values 2 from csv"
- digest: `"6fbad822cb9ae42deea1ede626890711"`, run name: "Default"

Run name may not be unique, but in examples below we going to use name just to improve readability.

To see the parameter value from particular model run:

```
http://localhost:4040/api/model/modelOne/run/Default/parameter/StartingSeed/value
```



```
[
 {
  "Dims": [],
  "IsNull": false,
  "Value": 1023,
  "SubId": 0
 }
]
```


Or

```
http://localhost:4040/api/model/modelOne/run/Default/parameter/baseSalary/value
```


```
[
  {
    "Dims": [],
    "IsNull": false,
    "Value": "Full",
    "SubId": 0
  }
]
```


Use model profile to supply parameter values

Profile is a set of key-value options, similar to ini-file, which can be used to run the model. Each profile can be identified by profile name. It may be more convenient to use profiles instead of ini-files because profiles are stored in database and you don't need to deal with multiple files in order to publish and run the model in cloud.

To create profile named `seed-1-base-full` with values of `StartingSeed` and `baseSalary` parameters :

```
curl -v -X PATCH -H "Content-Type: application/json" \
  "http://localhost:4040/api/model/modelOne/profile" \
  -d \
  '{ "Name": "seed-1-base-full",
     "Opts": {
       "OpenM.StartingSeed": "1023",
       "OpenM.baseSalary": "Full"
     }
  }'
```

Above curl command line is Linux specific, on Windows you must use ^ instead of \ for multi-line input and also double "quotes" and \" instead of single 'quotes'.

To view model profile:

```
http://localhost:4040/api/model/modelOne/profile/seed-1-base-full
```


```
{
  "Name": "seed-1-base-full",
  "Opts": {
    "OpenM.StartingSeed": "1023",
    "OpenM.baseSalary": "Full"
  }
}
```


To modify profile value:

```
curl -v -X POST http://localhost:4040/api/model/modelOne/profile/seed-1-base-full/key/Parameter.StartingSeed/value/4095
```

You can create multiple profiles similar to above in order to run the model with different `StartingSeed` and `baseSalary` parameter values:

```
modelOne -OpenM.Profile seed-1-base-full
modelOne -OpenM.Profile seed-1-base-part
modelOne -OpenM.Profile seed-2-base-full
modelOne -OpenM.Profile seed-2-base-part
```

It is the same as supply parameter values on command line:

```
modelOne -Parameter.StartingSeed 1023 -Parameter.baseSalary Full
modelOne -Parameter.StartingSeed 1023 -Parameter.baseSalary Part
modelOne -Parameter.StartingSeed 2047 -Parameter.baseSalary Full
modelOne -Parameter.StartingSeed 2047 -Parameter.baseSalary Part
```

Above model runs are using profile or command line values of `StartingSeed` and `baseSalary` and all other parameters are coming from "default" workset (default set of input parameters, a.k.a. default "scenario").

Simple way to create new workset (input set of parameters)

If you already run the model then database contains run results in output tables and copy of input parameters of that model run. We can use previous run parameters as "base" for our new workset, modify only some of it and run our model again.

1. To create `New-Set` of model parameters based on model run named "Default" with digest "6fbad822cb9ae42deea1ede626890711":

```
curl -v -X PUT \
  -F 'workset={
      "ModelName": "modelOne",
      "Name": "New-Set",
      "BaseRunDigest": "6fbad822cb9ae42deea1ede626890711",
      "Txt": [
        { "LangCode": "EN", "Descr": "My new set of input parameters" }
      ],
      "Param": [
        {
          "Name": "StartingSeed",
          "SubCount": 1,
          "Txt": [
            { "LangCode": "EN", "Note": "Starting seed new value" }
          ],
          "Value": [
            {"Dims": [], "IsNull": false, "Value": 8191, "SubId": 0}
          ]
        },
        {
          "Name": "ageSex",
          "SubCount": 1,
          "Txt": [],
          "Value": [
            {"Dims": ["10-20","M"], "IsNull": false, "Value": 0.1, "SubId": 0},
            {"Dims": ["10-20","F"], "IsNull": false, "Value": 0.2, "SubId": 0},
            {"Dims": ["20-30","M"], "IsNull": false, "Value": 0.3, "SubId": 0},
            {"Dims": ["20-30","F"], "IsNull": false, "Value": 0.4, "SubId": 0},
            {"Dims": ["30-40","M"], "IsNull": false, "Value": 0.5, "SubId": 0},
            {"Dims": ["30-40","F"], "IsNull": false, "Value": 0.6, "SubId": 0},
            {"Dims": ["40+","M"],   "IsNull": false, "Value": 0.7, "SubId": 0},
            {"Dims": ["40+","F"],   "IsNull": false, "Value": 0.8, "SubId": 0}
          ]
        }
      ]
    }' \
  http://localhost:4040/api/workset-create
```

That `New-Set` conatins new values for `StartingSeed` and `ageSex` parameters. All other input values are identical to previous "Default" model run input.

Each input set of model parameters (each workset) must have unique name. Different models can have worksets with same name, i.e. each model can have workset with name "Default". If workset with the same name `New-Set` already exist then this method return an error.

You don't have to create workset based on previous model run, you can omit `BaseRunDigest` and include all parameter values in the new workset. However it may be difficult for complex model with hundreds input parameters.

Advanced way to create new workset (input set of parameters) based on previous model run

If you already run the model then database contains run results in output tables and copy of input parameters of that model run. We can use previous run parameters as "base" for our new workset, modify only some of it and run our model again.

1. To create new `MyFirstSet` of model parameters based on model run named "Default" with digest "6fbad822cb9ae42deea1ede626890711":

```
curl -v -X PUT \
  -F 'workset={
     "ModelName": "modelOne",
     "Name": "MyFirstSet",
     "BaseRunDigest": "6fbad822cb9ae42deea1ede626890711",
     "Txt": [
       { "LangCode": "EN", "Descr": "My first set of input parameters" }
     ]
    }' \
  http://localhost:4040/api/workset-replace
```

That workset does not yet include any new parameter values, all input is identical to previous "Default" model run input. In order to modify parameter values we first need to copy into our new workset from any model run, any other workset or upload as csv-file.

2. Copy parameter `StartingSeed` value into `MyFirstSet` workset from `Default-4` model run:

```
curl -v -X PUT http://localhost:4040/api/model/modelOne/workset/MyFirstSet/copy/parameter/StartingSeed/from-run/Default-4
```

3. Copy parameter `baseSalary` value into `MyFirstSet` workset from `modelOne_other` workset:

```
curl -v -X PUT http://localhost:4040/api/model/modelOne/workset/MyFirstSet/copy/parameter/baseSalary/from-workset/modelOne_other
```

4. Upload parameter `ageSex` values into `MyFirstSet` workset from `my_age_sex.csv` csv file:

```
curl -v -X PATCH \
  -F 'workset={
      "ModelName": "modelOne",
      "Name": "MyFirstSet",
      "Param": [
        { "Name": "ageSex", "SubCount": 1 }
      ]
     }' \
  -F 'parameter-csv=@my_age_sex.csv;filename=ageSex.csv' \
  http://localhost:4040/api/workset-merge
```

where content of my_age_sex.csv is:

```
sub_id,dim0,dim1,param_value
0,10-20,M,11
0,10-20,F,12
0,20-30,M,13
0,20-30,F,14
0,30-40,M,15
0,30-40,F,16
0,40+,M,17
0,40+,F,18
```

It is also possible to modify some part of parameter values. For example, `ageSex` parameter above is 4*3 matrix and if want to modify values:

```
 [30-40, M] = 0.15
 [30-40, F] = 0.16
```

then:

```
curl -v -X PATCH -H "Content-Type: application/json" \
  http://localhost:4040/api/model/modelOne/workset/MyFirstSet/parameter/ageSex/new/value \
  -d '[
      {"Dims": ["30-40", "M"], "IsNull": false, "SubId":0, "Value": 0.15},
      {"Dims": ["30-40", "F"], "IsNull": false, "SubId":0, "Value": 0.16}
    ]'
```

Finally our "MyFirstSet" input set contains new values for 3 parameters: `StartingSeed, baseSalary, ageSex`, which different from previous "base run" parameters. And now we can **run our model with that new workset:**

```
modelOne -OpenM.SetName MyFirstSet
```

It is also possible to delete parameter from workset, delete entire workset in order to cleanup database and perform some other operations. Please see Oms: openM++ web-service API for details.

Create or modify modeling task

Modeling task consists of multiple sets of input data and can be run in batch mode. There is an example of modeling task at Run RiskPaths model from R page where we creating 800 sets of input data to study Childlessness by varying

- Age baseline for first union formation
- Relative risks of union status on first pregnancy After preparing such modeling task we can submit RiskPath model to high performance cluster (HPC) grid or in cloud where model will read 800 input sets and produce 800 model run outputs.

It is also possible to create or modify or delete modeling task without R, using Oms JSON web-service from programming language of your choice.

In order to do this we need first to prepare our input worksets as described above and after that we can create modeling task. For example, if we have two worksets: `MyFirstSet, MySecondSet` then we can create task:

```
curl -v -X PUT -H "Content-Type: application/json" \
  http://localhost:4040/api/task-new \
  -d '{
    "ModelName": "modelOne",
    "Name": "MyTask",
    "Txt": [{
        "LangCode": "EN",
        "Descr": "Task to vary 3 parameters",
        "Note": "Study effect of 3 parameters on output results"
      }
    ],
    "Set": [
      "MyFirstSet",
      "MySecondSet"
    ]
  }'
```

You can see the list of modeling tasks:

```
http://localhost:4040/api/model/modelOne/task-list
```

examine task metadata, input sets or task run history:

```
http://localhost:4040/api/model/modelOne/task/MyTask/text
http://localhost:4040/api/model/modelOne/task/MyTask/sets
http://localhost:4040/api/model/modelOne/task/MyTask/runs
```

It is also possible to delete or modify task. For example, if you want to add `MyThirdSet` set of parameters to the task above:

```
curl -v -X PATCH -H "Content-Type: application/json" \
  http://localhost:4040/api/task \
  -d '{
    "ModelName": "modelOne",
    "Name": "MyTask",
    "Set": [
      "MyThirdSet"
    ]
  }'
```

After that task will contain 3 input worksets:

```
http://localhost:4040/api/model/modelOne/task/MyTask/sets
```

```
{
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "MyTask",
  "Txt": [],
  "Set": [
    "MyFirstSet",
    "MySecondSet",
    "MyThirdSet"
  ],
  "TaskRun": []
}
```

Now you can run the model with that task:

```
modelOne -OpenM.SubValues 16 -OpenM.TaskName MyTask -OpenM.TaskRunName MyTask-sub16
```

and examine history of modeling task run:

```
http://localhost:4040/api/model/modelOne/task/MyTask/runs
```

```
{
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "MyTask",
  "Txt": [],
  "Set": [],
  "TaskRun": [
    {
      "Name": "MyTask-sub16",
      "SubCount": 16,
      "CreateDateTime": "2019-01-16 04:38:53.0298",
      "Status": "s",
      "UpdateDateTime": "2019-01-16 04:38:53.0461",
      "TaskRunSet": [
        {
          "Run": {
            "Name": "MyTask_sub16_MyFirstSet_2019_01_16_04_38_53_0304_111",
            "SubCompleted": 16,
            "CreateDateTime": "2019-01-16 04:38:53.0304",
            "Status": "s",
            "Digest": "1cece5a11d522b6225d7f9cb5afda39a"
          },
          "SetName": "MyFirstSet"
        },
        {
          "Run": {
            "Name": "MyTask_sub16_MySecondSet_2019_01_16_04_38_53_0357_112",
            "SubCompleted": 16,
            "CreateDateTime": "2019-01-16 04:38:53.0357",
            "Status": "s",
            "Digest": "4a55cd6614f8f7be439c0776b2a473ab"
          },
          "SetName": "MySecondSet"
        },
        {
          "Run": {
            "Name": "MyTask_sub16_MyThirdSet_2019_01_16_04_38_53_0410_113",
            "SubCompleted": 16,
            "CreateDateTime": "2019-01-16 04:38:53.0410",
            "Status": "s",
            "Digest": "d112237f501317422943880eca54d07b"
          },
          "SetName": "MyThirdSet"
        }
      ]
    }
  ]
```

Oms: Cloud and model runs queue

OpenM++ web-service (oms) can provide basic computational resources management for your local computer or cluster of servers on local network or in cloud. It can manage model runs queue if your computational resources (CPU and memory) are limited and also can automatically start and stop cloud servers.

Examples below assuming you are familiar with basics of Oms: openM++ web-service.

If you want to have model runs queue, or using openM++ in cloud and want automatically scale up and down cloud resources, e.g. start and stop virtual machines for model runs then start `oms` with job control option:

```
oms -oms.JobDir job
```

Following directory structure expected:

```
./       -> oms "root" directory, by default it is current directory
    html/    -> web-UI directory with HTML, js, css, images...
    etc/     -> config files directory, contain template(s) to run models
    log/     -> recommended log files directory
    models/
          bin/  -> default model.exe and model.sqlite directory
          log/  -> default directory for models run log files
            home/ -> user personal home directory
                io/download  -> user directory for download files
              io/upload    -> user directory to upload files
    job/  -> model run jobs control directory
          job.ini   -> job control settings
          active/   -> active model run state files
          history/  -> model run history files
          queue/    -> model run queue files
          state/    -> jobs state and computational servers state files
               jobs.queue.paused  -> if such file exists then jobs queue is paused
```

Model runs queue and computational resources (servers, nodes, clusters)

By default `oms` assumes:

- all models are running on `localhost`
- there are no limits on CPU cores or memory usage

Model run queue on local computer

You can create model run queue on your local computer by setting a limit on number of CPU cores available. To do it modify `job.ini` file in a `job` directory, for example:

```
[Common]
LocalCpu      = 8       ; localhost CPU cores limit, localhost limits are applied only to non-MPI jobs
LocalMemory   = 0       ; gigabytes, localhost memory limit, zero means no limits
```

You don't have to set memory limits until model run memory requirements are known.

CPU cores which are you limiting in `job.ini` does not need to be an actual cores. You can have 8 cores on your PC and set `LocalCpu = 16` which allow 200% overload and may significantly slow down your local machine. Or if you set `LocalCpu = 4` then your models would be able to use only half of actual cores.

LAN: front-end server and back-end cluster of servers

Example of local network (LAN) cluster:

- small front-end server with 4 cores
- 4 back-end servers: cpc-1, cpc-2, cpc-3, cpc-4 with 16 cores each

```
[Common]
LocalCpu      = 4       ; localhost CPU cores limit, localhost limits are applied only to non-MPI jobs
LocalMemory   = 0       ; gigabytes, localhost memory limit, zero means no limits
MpiMaxThreads = 8       ; max number of modelling threads per MPI process
MaxErrors     = 10      ; errors threshold for compute server or cluster

Servers   = cpc-1, cpc-2, cpc-3, cpc-4      ; computational servers or clusters

[cpc-1]
Cpu       = 16         ; default: 1 CPU core
Memory    = 0          ; zero means no limits

[cpc-2]
Cpu       = 16         ; default: 1 CPU core
Memory    = 0          ; zero means no limits

[cpc-3]
Cpu       = 16         ; default: 1 CPU core
Memory    = 0          ; zero means no limits

[cpc-4]
Cpu       = 16         ; default: 1 CPU core
Memory    = 0          ; zero means no limits

; OpenMPI hostfile (on Linux)
;
; cpm   slots=1 max_slots=1
; cpc-1 slots=2
; cpc-3 slots=4
;
[hostfile]
HostFileDir = models/log
HostName = @-HOST-@
CpuCores = @-CORES-@
RootLine = cpm slots=1 max_slots=1
HostLine = @-HOST-@ slots=@-CORES-@

; MS-MPI machinefile (on Windows with Microsoft MPI)
;
; cpm:1
; cpc-1:2
; cpc-3:4
;
; [hostfile]
; HostFileDir = models\log
; HostName = @-HOST-@
; CpuCores = @-CORES-@
; RootLine = cpm:1
; HostLine = @-HOST-@:@-CORES-@
```

Based on `job.ini` above oms will create MPI `hostfile` with back-end servers assignment for each particular model run. In order to use that `hostfile` you should modify model run template(s) in openM++ `etc/` directory. For example on Linux with openMPI:

```
{{/*
oms web-service:
  Template to run modelName_mpi executable on Linux using OpenMPI

It is not recommended to use root process for modelling

Oms web-service using template for exec.Command(exeName, Args...):
  - skip empty lines
  - substitute template arguments
  - first non-empty line is a name of executable to run
  - each other line is a command line argument for executable

Arguments of template:
  ModelName string        // model name
  ExeStem   string        // base part of model exe name, usually modelName
  Dir       string        // work directory to run the model
  BinDir    string        // bin directory where model exe is located
  MpiNp     int           // number of MPI processes
  HostFile  string        // if not empty then path to hostfile
  Args      []string      // model command line arguments
  Env       map[string]string // environment variables to run the model

Example of result:

  mpirun --hostfile host.ini --bind-to none --oversubscribe -wdir models/bin -x key=value ./modelName_mpi -OpenM.LogToFile false

*/}}

mpirun
--bind-to
none
--oversubscribe
{{with .HostFile}}
--hostfile
{{.}}
{{end}}
{{with .Dir}}
-wdir
{{.}}
{{end}}
{{range $key, $val := .Env}}
-x
{{$key}}={{$val}}
{{end}}
{{.BinDir}}/{{.ExeStem}}_mpi
{{range .Args}}
{{.}}
{{end}}
```

Note: If you are using OpenMPI then it is a good idea to have `--oversubscribe --bind-to none` as above in order to avoid MPI models run failure or performance degradation.

If you are using Microsoft MPI on Windows servers then modify `etc\` model template file(s) to have it similar to:

```
{{/*
oms web-service:
  Template to run modelName_mpi.exe on Windows Microsoft MPI using machinefile

To use this template rename it into:
  mpi.ModelRun.template.txt

Oms web-service using template for exec.Command(exeName, Args...):
  - skip empty lines
  - substitute template arguments
  - first non-empty line is a name of executable to run
  - each other line is a command line argument for executable

Arguments of template:
  ModelName string            // model name
  ExeStem   string            // base part of model exe name, usually modelName
  Dir       string            // work directory to run the model
  BinDir    string            // bin directory where model exe is located
  DbPath    string            // absolute path to sqlite database file: models/bin/model.sqlite
  MpiNp     int               // number of MPI processes
  HostFile  string            // if not empty then path to hostfile
  Args      []string          // model command line arguments
  Env       map[string]string // environment variables to run the model

Example of result:
  mpiexec -machinefile hosts.ini -wdir models\bin -env key value ..\bin\modelName_mpi -OpenM.LogToFile false

*/}}

mpiexec
{{with .HostFile}}
-machinefile
{{.}}
{{end}}
{{with .Dir}}
-wdir
{{.}}
{{end}}
{{range $key, $val := .Env}}
-env
{{$key}}
{{$val}}
{{end}}
{{.BinDir}}\{{.ExeStem}}_mpi
{{range .Args}}
{{.}}
{{end}}
```

Cloud auto scaling: automatically start and stop servers

Use `oms` jobs control abilities to organize model runs queue and, if required, automatically scale up down cloud resources, e.g.: start and stop virtual machines or nodes.

For example, if you want to have two users: Alice and Bob who are running models then start `oms` as:

```
bin/oms -l localhost:4050 -oms.RootDir alice -oms.Name alice -ini oms.ini
bin/oms -l localhost:4060 -oms.RootDir bob   -oms.Name bob   -ini oms.ini
```

where content of `oms.ini` is:

```
[oms]
JobDir        = ../job
EtcDir        = ../etc
HomeDir       = models/home
AllowDownload = true
AllowUpload   = true
LogRequest    = true

[OpenM]
LogFilePath      = log/oms.log
LogToFile        = true
LogUseDailyStamp = true
LogToConsole     = false
```

Above assume following directory structure:

```
./    -> current directory
    bin/
        oms    -> oms web service executable, on Windows: `oms.exe`
        dbcopy -> dbcopy utility executable, on Windows: `dbcopy.exe`
    html/    -> web-UI directory with HTML, js, css, images...
    etc/     -> config files directory, contain template(s) to run models
    log/     -> recommended log files directory
    alice/   -> user Alice "root" directory
        models/
              bin/  -> Alice's model.exe and model.sqlite directory
              log/  -> Alice's directory for models run log files
                home/ -> Alice's personal home directory
                  io/download  -> Alice's directory for download files
                io/upload    -> Alice's directory to upload files
    bob/     -> user Bob "root" directory
        models/
              bin/  -> Bob's model.exe and model.sqlite directory
              log/  -> Bob's directory for models run log files
                home/ -> Bob's personal home directory
                  io/download  -> Bob's directory for download files
                io/upload    -> Bob's directory to upload files
    job/  -> model run jobs control directory, it must be shared between all users
          job.ini   -> (optional) job control settings
          active/   -> active model run state files
          history/  -> model run history files
          queue/    -> model run queue files
          state/    -> jobs state and computational servers state files
               jobs.queue.paused  -> if such file exists then jobs queue is paused
```

You don't have to follow that directory structure, it is flexible and can be customized through `oms` run options.

IMPORTANT: Job directory must be in a SHARED location and accessible to all users who are using the same queue and the same computational resources (servers, nodes, clusters).

You don't need to create OS users, e.g. Alice and Bob does not need a login accounts on your server (cloud, Active Directory, etc.). All you need is to setup some authentication mechanism and reverse proxy which would allow Alice to access `localhost:4050` and Bob `localhost:4060` on your front-end. Actual OS user can have any name, e.g. `oms`:

```
sudo -u oms OM_ROOT=/shared/alice bash -c 'source ~/.bashrc; bin/oms -l localhost:4050 -oms.RootDir alice -oms.Name alice -ini oms.ini &'
sudo -u oms OM_ROOT=/shared/bob   bash -c 'source ~/.bashrc; bin/oms -l localhost:4060 -oms.RootDir bob   -oms.Name bob   -ini oms.ini &'
```

Google cloud: front-end server and and auto scale of multiple back-end servers

There is a small front-end server with 4 cores and 4 back-end servers: cpc-1, cpc-2, cpc-3, cpc-4 with 16 cores each. You are using public cloud and want to pay only for actual usage of back end servers:

- server(s) must be started automatically when user (Alice or Bob) want to run the model;
- server(s) must stop after model run completed to reduce cloud cost

Scripts below are also available at our GitHub↗

```
[Common]
LocalCpu      = 4       ; localhost CPU cores limit, localhost limits are applied only to non-MPI jobs
LocalMemory   = 0       ; gigabytes, localhost memory limit, zero means no limits
MpiMaxThreads = 8       ; max number of modelling threads per MPI process
MaxErrors     = 10      ; errors threshold for compute server or cluster
IdleTimeout   = 900     ; seconds, idle time before stopping server or cluster
StartTimeout  = 180     ; seconds, max time to start server or cluster
StopTimeout   = 180     ; seconds, max time to stop server or cluster

Servers   = cpc-1, cpc-2, cpc-3, cpc-4  ; computational servers or clusters

StartExe  = /bin/bash                   ; default executable to start server
StopExe   = /bin/bash                   ; default executable to stop server
ArgsBreak = -@-                         ; arguments delimiter in StartArgs or StopArgs line
                                        ; delimiter can NOT contain ; or # chars, which are reserved for # comments
                                        ; it can be any other delimiter of your choice, e.g.: +++
; StartArgs = ../etc/compute-start.sh    ; default command line arguments to start server, server name will be appended
; StopArgs  = ../etc/compute-stop.sh     ; default command line arguments to start server, server name will be appended

[cpc-1]
Cpu       = 16          ; default: 1 CPU core
Memory    = 0           ; zero means no limits
StartArgs = ../etc/compute-start-4.sh-@-us-zone-b-@-cpc-1
StopArgs  = ../etc/compute-stop-4.sh-@-us-zone-b-@-cpc-1

[cpc-2]
Cpu       = 16          ; default: 1 CPU core
Memory    = 0           ; zero means no limits
StartArgs = ../etc/compute-start-4.sh-@-us-zone-c-@-cpc-2
StopArgs  = ../etc/compute-stop-4.sh-@-us-zone-c-@-cpc-2

[cpc-3]
Cpu       = 16          ; default: 1 CPU core
Memory    = 0           ; zero means no limits
StartArgs = ../etc/compute-start-4.sh-@-us-zone-d-@-cpc-3
StopArgs  = ../etc/compute-stop-4.sh-@-us-zone-d-@-cpc-3

[cpc-4]
Cpu       = 16          ; default: 1 CPU core
Memory    = 0           ; zero means no limits
StartArgs = ../etc/compute-start-4.sh-@-us-zone-a-@-cpc-4
StopArgs  = ../etc/compute-stop-4.sh-@-us-zone-a-@-cpc-4

; OpenMPI hostfile
;
; cpm   slots=1 max_slots=1
; cpc-1 slots=2
; cpc-3 slots=4
;
[hostfile]
HostFileDir = models/log
HostName = @-HOST-@
CpuCores = @-CORES-@
RootLine = cpm slots=1 max_slots=1
HostLine = @-HOST-@ slots=@-CORES-@

; MS-MPI machinefile (on Windows with Microsoft MPI)
;
; cpm:1
; cpc-1:2
; cpc-3:4
;
; [hostfile]
; HostFileDir = models\log
; HostName = @-HOST-@
; CpuCores = @-CORES-@
; RootLine = cpm:1
; HostLine = @-HOST-@:@-CORES-@
```

Oms is using `StartExe` and `StartArgs` in order to start each server. On Linux result of above `job.ini` is:

```
/bin/bash etc/compute-start.sh cpc-1
```

On Windows you can use `cmd` or PowerShell in order to control servers. Related part of `job.ini` can look like:

```
StartExe  = cmd                        ; default executable to start server
StartArgs = /C-@-etc\compute-start.bat  ; default command line arguments to start server, server name will be appended
StopExe   = cmd                        ; default executable to stop server
StopArgs  = /C-@-etc\compute-stop.bat   ; default command line arguments to start server, server name will be appended
```

which result in following command to start server:

```
cmd /C etc\compute-start.bat cpc-1
```

Start and stop scripts can look like (Google cloud version):

```
#!/bin/bash
#
# start computational server, run as:
#
# sudo -u $USER-NAME compute-start.sh host-name

srv_zone="us-zone-b"
srv_name="$1"

if [ -z "$srv_name" ] || [ -z "$srv_zone" ] ;
then
  echo "ERROR: invalid (empty) server name or zone: $srv_name $srv_zone"
  exit 1
fi

gcloud compute instances start $srv_name --zone $srv_zone
status=$?

if [ $status -ne 0 ];
then
  echo "ERROR $status at start of: $srv_name"
  exit $status
fi

# wait until MPI is ready

for i in 1 2 3 4; do

  sleep 10

  echo "[$i] mpirun -n 1 -H $srv_name hostname"

  mpirun -n 1 -H $srv_name hostname
  status=$?

  if [ $status -eq 0 ] ; then break; fi
done

if [ $status -ne 0 ];
then
  echo "ERROR $status from MPI at start of: $srv_name"
  exit $status
fi

echo "Start OK: $srv_name"
```

```bash
#!/bin/bash
#
# stop computational server, run as:
#
# sudo -u $USER-NAME compute-stop.sh host-name

# set -e

srv_zone="us-zone-b"
srv_name="$1"

if [ -z "$srv_name" ] || [ -z "$srv_zone" ] ;
then
  echo "ERROR: invalid (empty) server name or zone: $srv_name $srv_zone"
  exit 1
fi

for i in 1 2 3 4 5 6 7; do

  gcloud compute instances stop $srv_name --zone $srv_zone
  status=$?

  if [ $status -eq 0 ] ; then break; fi

  sleep 10
done

if [ $status -ne 0 ];
then
  echo "ERROR $status at stop of: $srv_name"
  exit $status
fi

echo "Stop OK: $srv_name"
```

Azure cloud: front-end server and and auto scale of multiple back-end servers

There is a small front-end server with 4 cores and 2 back-end servers: dc1, dc2 with 4 cores each. You are using public cloud and want to pay only for actual usage of back end servers:

- server(s) must be started automatically when user (Alice or Bob) want to run the model;
- server(s) must stop after model run completed to reduce cloud cost

Scripts below are also available at our GitHub↗

```
[Common]
LocalCpu      = 4     ; localhost CPU cores limit, localhost limits are applied only to non-MPI jobs
LocalMemory   = 0     ; gigabytes, localhost memory limit, zero means unlimited
MpiMaxThreads = 8     ; max number of modelling threads per MPI process
MaxErrors     = 10    ; errors threshold for compute server or cluster
IdleTimeout   = 900   ; seconds, idle time before stopping server or cluster
StartTimeout  = 90    ; seconds, max time to start server or cluster
StopTimeout   = 90    ; seconds, max time to stop server or cluster

Servers   = dc1, dc2     ; computational servers or clusters for MPI jobs

StartExe  = /bin/bash                      ; default executable to start server
StopExe   = /bin/bash                      ; default executable to stopt server
StartArgs = ../etc/az-start.sh-@-dm_group   ; default command line arguments to start server, server name will be appended
StopArgs  = ../etc/az-stop.sh-@-dm_group    ; default command line arguments to start server, server name will be appended

ArgsBreak = -@-                   ; arguments delimiter in StartArgs or StopArgs line
                                  ; delimiter can NOT contain ; or # chars, which are reserved for # comments
                                  ; it can be any other delimiter of your choice, e.g.: +++

[dc1]
Cpu    = 4    ; default: 1 CPU core
Memory = 0

[dc2]
Cpu    = 4    ; default: 1 CPU core
Memory = 0

; OpenMPI hostfile
;
; dcm slots=1 max_slots=1
; dc1 slots=2
; dc2 slots=4
;
[hostfile]
HostFileDir = models/log
HostName    = @-HOST-@
CpuCores    = @-CORES-@
RootLine    = dm slots=1 max_slots=1
HostLine    = @-HOST-@ slots=@-CORES-@
```

Oms is using `StartExe` and `StartArgs` in order to start each server. On Linux result of above `job.ini` is similar to:

```
/bin/bash etc/az-start.sh dm_group dc1
```

Start and stop scripts can look like (Azure cloud version):

```bash
#!/bin/bash
#
# start Azure server, run as:
#
# sudo -u $USER-NAME az-start.sh resource-group host-name

# set -e

res_group="$1"
srv_name="$2"

if [ -z "$srv_name" ] || [ -z "$res_group" ] ;
then
  echo "ERROR: invalid (empty) server name or resource group: $srv_name $res_group"
  exit 1
fi

# login

az login --identity
status=$?

if [ $status -ne 0 ];
then
  echo "ERROR $status from az login at start of: $res_group $srv_name"
  exit $status
fi

# Azure VM start

az vm start -g "$res_group" -n "$srv_name"
status=$?

if [ $status -ne 0 ];
then
  echo "ERROR $status at: az vm start -g $res_group -n $srv_name"
  exit $status
fi

# wait until MPI is ready

for i in 1 2 3 4 5; do

  sleep 10

  echo "[$i] mpirun -n 1 -H $srv_name hostname"

  mpirun -n 1 -H $srv_name hostname
  status=$?

  if [ $status -eq 0 ] ; then break; fi
done

if [ $status -ne 0 ];
then
  echo "ERROR $status from MPI at start of: $srv_name"
  exit $status
fi

echo "Start OK: $srv_name"
```

```bash
#!/bin/bash
#
# stop Azure server, run as:
#
# sudo -u $USER-NAME az-stop.sh resource-group host-name

# set -e

res_group="$1"
srv_name="$2"

if [ -z "$srv_name" ] || [ -z "$res_group" ] ;
then
  echo "ERROR: invalid (empty) server name or resource group: $srv_name $res_group"
  exit 1
fi

# login

az login --identity
status=$?

if [ $status -ne 0 ];
then
  echo "ERROR $status from az login at start of: $res_group $srv_name"
  exit $status
fi

# Azure VM stop

for i in 1 2 3 4; do

  az vm deallocate -g "$res_group" -n "$srv_name"

  if [ $status -eq 0 ] ; then break; fi

  sleep 10
done

if [ $status -ne 0 ];
then
  echo "ERROR $status at stop of: $srv_name"
  exit $status
fi

echo "Stop OK: $srv_name"
```

Linux cluster in cloud

Security consideration:

In wiki I am describing the most simple but least secure configuration, for your production environment you may want to:

- use a separate web front-end server, separate `oms` control server with firewall in between
- never use front-end web-server OS user as `oms` control server OS user
- do not use the same OS user, like `oms`, but create a different for each of your model users, like Alice and Bob in example above.

Of course web front-end UI of your production environment must be protected by `https://` with proper authentication and authorization. All that is out of scope of our wiki, please consult your organization security guidelines for it.

Also I am not describing here how to configure web-servers, how to create reverse proxy, install SSL certificates, etc. There are a lot of great materials on those topics around, just please think about security in a first place.

Cloud examples here assume Debian or Ubuntu Linux servers setup, you can use it for RedHat Linux with minimal adjustment. OpenM++ do support Microsoft Windows clusters, but configuring it is a more complex task and out of scope for that wiki.

Our simple cluster consist of from-end web-UI server with host name `dm` and multiple back-end computational servers: `dc1, dc2,...`.

Front-end server OS setup

Front-end `dm` server must have some web-server installed, Apache or nginx for example, static IP and DNS records for your domain.

Choose Debian-11, Ubuntu 22.04 or RedHat 9 (Rocky, AlmaLinux) as your base system and create `dm` cloud virtual machine, at least 4 cores recommended. We will create two disks on `dm`: boot disk and fast SSD data disk where all users data and models are stored.

Set timezone, install openMPI and (optional) SQLite:

```
sudo timedatectl set-timezone America/Toronto

sudo apt-get install openmpi-bin
sudo apt-get install sqlite3

# check result:
mpirun hostname -A
```

Create and mount on `/mirror` SSD data disk to store all users data and models:

```
# init new SSD, use lsblk to find which /dev it is
lsblk

sudo mkfs.ext4 -m 0 -E lazy_itable_init=0,lazy_journal_init=0,discard /dev/sda

sudo mkdir /mirror
sudo mount -o discard,defaults /dev/sda /mirror

# check results:
ls -la /mirror

# add new disk to fstab, mount by UUID:
sudo blkid /dev/sda
sudo nano /etc/fstab

# add your UUID mount:
UUID=98765432-d09a-4936-b85f-a61da123456789 /mirror ext4 discard,defaults 0 2
```

Create NFS shares:

```
sudo mkdir -p /mirror/home
sudo mkdir -p /mirror/data

sudo apt install nfs-kernel-server

# add shares into exports:
sudo nano /etc/exports

# export user homes and data, data can be exported read-only, rw is not required
/mirror/home *(rw,sync,no_root_squash,no_subtree_check)
/mirror/data *(rw,sync,no_root_squash,no_subtree_check)

sudo systemctl restart nfs-kernel-server

# check results:
/sbin/showmount -e dm

systemctl status nfs-kernel-server
```

Create 'oms' service account, login disabled. I am using 1108 as user id and group id, but it is an example only and 1108 have no special meaning:

```
export OMS_UID=1108
export OMS_GID=1108

sudo addgroup --gid $OMS_GID oms
sudo adduser --home /mirror/home/oms --disabled-password --gecos "" --gid $OMS_GID -u $OMS_UID oms

sudo chown -R oms:oms /mirror/data

# increase stack size for models to 65 MB = 65536

sudo -u oms nano /mirror/home/oms/.bashrc

# ~/.bashrc: executed by bash(1) for non-login shells.
# openM++
# some models require stack size:
#
ulimit -S -s 65536

#
# end of openM++
```

Password-less ssh for `oms` service account:

```
sudo su -l oms
cd ~

mkdir .ssh

ssh-keygen -f .ssh/id_rsa -t rsa -N '' -C oms

# create .ssh/config with content below:
nano .ssh/config

Host *
   StrictHostKeyChecking no
   UserKnownHostsFile /dev/null
   LogLevel ERROR

cp -p .ssh/id_rsa.pub .ssh/authorized_keys

chmod 700 .ssh
chmod 600 .ssh/id_rsa
chmod 644 .ssh/id_rsa.pub
chmod 644 .ssh/config
chmod 644 .ssh/authorized_keys

exit    # logout from 'oms' user

# check ssh for oms user, it should work without any prompts, without any Yes/No questions:

sudo -u oms ssh dm
```

Check openMPI under 'oms' service account:

```
sudo -u oms mpirun hostname
sudo -u oms mpirun -H dm hostname
```

Done with `dm` server OS setup, reboot it and start `dc1, dc2,...` creating back-end servers.

Back-end computational servers setup

I am describing it for `dc1`, assuming you will create base image from it and use for all other back-end servers. On Azure it is make sense to create virtual machine scale set instead of individual servers.

Choose Debian-11, Ubuntu 22.04 or RedHat 9 (Rocky, AlmaLinux) as your base system and create `dc1` cloud virtual machine, at least 16 cores recommended. It does not require a fast SSD, use regular small HDD because there are no model data stored in back-end, it is only OS boot disk, nothing else. Back-end servers should not be visible from the internet, it should be visible only from front-end `dm` server.

Set timezone and install openMPI::

```
sudo timedatectl set-timezone America/Toronto

sudo apt-get install openmpi-bin

# check result:
mpirun hostname -A
```

Mount NFS shares from `dm` server:

```
sudo mkdir -p /mirror/home
sudo mkdir -p /mirror/data

sudo apt install nfs-common

/sbin/showmount -e dm

sudo mount -t nfs dm:/mirror/home /mirror/home
sudo mount -t nfs dm:/mirror/data /mirror/data

systemctl status mirror-home.mount
systemctl status mirror-data.mount

# if above OK then add nfs share mounts into fstab:

sudo nano /etc/fstab

# fstab records:
dm:/mirror/home /mirror/home nfs defaults 0 0
dm:/mirror/data /mirror/data nfs defaults 0 0

# (optional) reboot node and make sure shares are mounted:

systemctl status mirror-home.mount
systemctl status mirror-data.mount
```

Create 'oms' service account, login disabled. It must have exactly the same user id and group id as `oms` user on `dm`, I am using 1108 as an example:

```
export OMS_UID=1108
export OMS_GID=1108

sudo /sbin/addgroup --gid $OMS_GID oms
sudo adduser --no-create-home --home /mirror/home/oms --disabled-password --gecos "" --gid $OMS_GID -u $OMS_UID oms

# check 'oms' sevice account access to shared files:

sudo -u oms -- ls -la /mirror/home/oms/.ssh/
```

Optional: if you are using Azure virtual machine scale set then cloud.init config can be:

```
#cloud-config
#
runcmd:
 - addgroup --gid 1108 oms
 - adduser --no-create-home --home /mirror/home/oms --disabled-password --gecos "" --gid 1108 -u 1108 oms
```

Check openMPI under 'oms' service account:

```
sudo -u oms mpirun hostname
sudo -u oms mpirun -H dc1 hostname
sudo -u oms mpirun -H dm hostname
```

Done with `dc1` OS setup, clone it for all other back-end servers. After you created all back-end servers check openMPI from entire cluster, for example:

```
sudo -u oms mpirun -H dm,dc1,dc2,dc3,dc4,dc5,dc6,dc7,dc8,dc9,dc10 hostname
```

Now login back to your `dm` front-end and create standard openM++ directory structure at `/mirror/data/`, copy models, create user directories as it is described for "users" Alice and Bob above. Bob and Alice are your model users, they should not have OS login, user `oms` with disabled login is used to run the models on behalf of Alice and Bob. I would also recommend to have at least one "user" for your own tests, to verify system status and test and run the models when you publish it. For that I am usually creating "user" `test`.

```
/mirror/data/
    bin/
        oms     -> oms web service executable
        dbcopy -> dbcopy utility executable
    html/    -> web-UI directory with HTML, js, css, images...
    etc/     -> config files directory, contain template(s) to run models
    log/     -> recommended log files directory
    alice/   -> user Alice "root" directory
        models/
              bin/  -> Alice's model.exe and model.sqlite directory
              log/  -> Alice's directory for models run log files
                home/ -> Alice's personal home directory
                  io/download  -> Alice's directory for download files
                io/upload    -> Alice's directory to upload files
    bob/     -> user Bob "root" directory
        models/
              bin/  -> Bob's model.exe and model.sqlite directory
              log/  -> Bob's directory for models run log files
                home/ -> Bob's personal home directory
                  io/download  -> Bob's directory for download files
                io/upload    -> Bob's directory to upload files
    job/  -> model run jobs control directory, it must be shared between all users
          job.ini   -> (optional) job control settings
          active/   -> active model run state files
          history/  -> model run history files
          queue/    -> model run queue files
          state/    -> jobs state and computational servers state files
    oms/    -> oms init.d files, see examples on our GitHub
    oms.ini -> oms config, see content above
    test/   -> user test "root" directory, for admin internal use
            -> .... user test subdirectories here
```

Above there is also `oms/` directory with `init.d` files to restart `oms` when front-end `dm` server is rebooted. You can find examples of it at our GitHub↗.

Use R to save output table into CSV file

Use R to save output table into CSV file

It is a convenient to use GNU R to prepare model parameters and analyze output values. There are two different R APIs which we can use for openM++ models:

- openMpp package: simple and convenient specially for desktop users, upstream and downstream analysis;
- `oms` JSON web-service API: preferable choice to run models on computational clusters and in cloud.

Below is an example how to use `oms` JSON web-service to read output table values from multiple model runs and save it into CSV file. In that example we are reading `RiskPaths` model output table `T04_FertilityRatesByAgeGroup` values from 3 model runs: `"New 123,000 cases", "New 456,000 cases", "New 789,000 cases"` and saving it into `T04_FertilityRatesByAgeGroup.csv`.

R script

```
#
# Read table values from multiple model runs and save it as TableName.csv
#
library("jsonlite")
library("httr")

# Include openM++ helper functions from your $HOME directory
#
source("~/omsCommon.R")

#
# Model digest of RiskPaths version 3.0.0.0: "d90e1e9a49a06d972ecf1d50e684c62b"
# We MUST use model digest if there are multiple versions of the model published.
# We can use model name if only single version of the model is published.
#
md <- "d90e1e9a49a06d972ecf1d50e684c62b"

# oms web-service URL from file: ~/oms_url.txt
#
apiUrl <- getOmsApiUrl()

# model runs can be identified by digest, by run stamp or by run name
# run digest is unique and it preferable way to identify model run
# run names are user friendly may not be unique
#
runNames <- c(
  "New 123,000 cases",
  "New 456,000 cases",
  "New 789,000 cases"
  )

# combine all run results and write it into T04_FertilityRatesByAgeGroup.csv
#
tableName <- "T04_FertilityRatesByAgeGroup"

allCct <- NULL

nRuns <- length(runNames)

for (k in 1:nRuns)
{
  cct <- read.csv(paste0(
    apiUrl, "model/", md, "/run/", URLencode(runNames[k], reserved = TRUE), "/table/", tableName, "/expr/csv"
    ))
  cct$RunName <- runNames[k]

  allCct <- rbind(allCct, cct)
}

write.csv(allCct, paste0(tableName, ".csv"), row.names = FALSE)
```

Use R to save output table into Excel

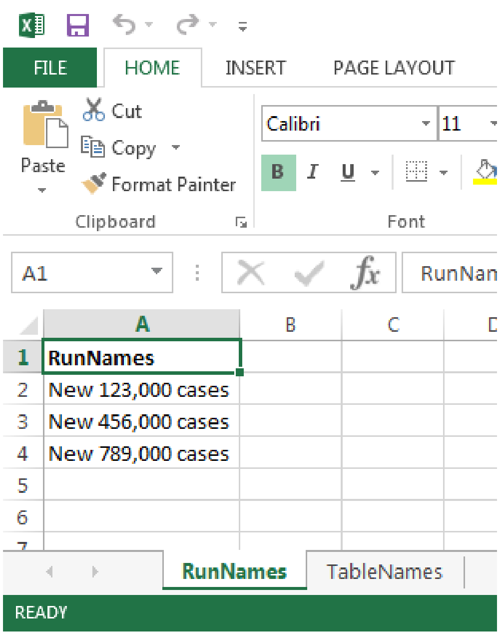
Use R to save output table into Excel

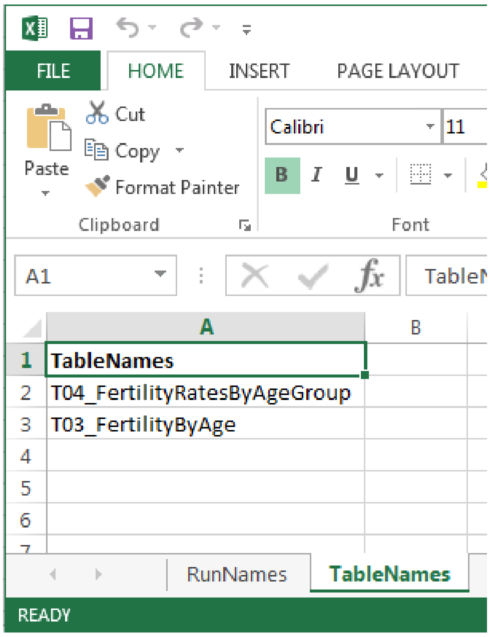
It is a convenient to use GNU R to prepare model parameters and analyze output values. There are two different R APIs which we can use for openM++ models:

- openMpp package: simple and convenient specially for desktop users, upstream and downstream analysis;
- `oms` JSON web-service API: preferable choice to run models on computational clusters and in cloud.

Below is an example how to use `oms` JSON web-service to read multiple output table values from multiple model runs and save it into XLSX file:

- using `RiskPaths` demo model
- reading model run names and output table names from input Excel file as on screenshots below
- for each table retriving output values for all model runs
- retrieving model runs metadata: run name, description, notes, date and time
- retrieving output tables metadata: name, description and notes
- saving each table output values as separate Excel workbook sheet
- saving all model runs metadata and tables metadata as separate sheets





R script

```
#
# Read multiple tables from multiple model runs and save it as XLSX file
# Also save model runs meatadata and tables metadata (name, description, notes) into .csv files
# Model run names and table names are coming from another input XLSX file
#
library("jsonlite")
library("httr")
library("readxl")
library("writexl")

# Include openM++ helper functions from your $HOME directory
#
source("~/omsCommon.R")

#
# Model digest of RiskPaths version 3.0.0.0: "d90e1e9a49a06d972ecf1d50e684c62b"
# We MUST use model digest if there are multiple versions of the model published.
# We can use model name if only single version of the model is published.
#
md <- "d90e1e9a49a06d972ecf1d50e684c62b"

# oms web-service URL from file: ~/oms_url.txt
#
```

```
#
apiUrl <- getOmsApiUrl()

# model runs can be identified by digest, by run stamp or by run name
# run digest is unique and it preferable way to identify model run
# run names are user friendly may not be unique
#
# read model run names from some XLSX file,
#   it must have sheet name = "RunNames" with A column "RunNames"
#
rn <- read_xlsx(
  "model-runs-to-read-and-tables-to-read.xlsx",
  sheet = "RunNames",
  col_types = "text"
  )

# read table names from some XLSX file,
#   it must have sheet name = "TableNames" with A column "TableNames"
#
tn <- read_xlsx(
  "model-runs-to-read-and-tables-to-read.xlsx",
  sheet = "TableNames",
  col_types = "text"
  )

# get table information
#
rsp <- GET(paste0(
    apiUrl, "model/", md, "/text"
  ))
if (http_type(rsp) != 'application/json') {
  stop("Failed to get first model info")
}
jr <- content(rsp)
tTxt <- jr$TableTxt

tableInfo <- data.frame()

for (t in tTxt) {
  for (tbl in tn$TableNames)
  {
    if (t$Table$Name == tbl) {
      ti <- data.frame(
          TableName = tbl,
          TableDescription = t$TableDescr,
          TableNotes = t$TableNote
        )
      tableInfo <- rbind(tableInfo, ti)
      break
    }
  }
}

# get run information
#
runInfo <- data.frame()

for (run in rn$RunNames)
{
  rsp <- GET(paste0(
      apiUrl, "model/", md, "/run/", URLencode(run, reserved = TRUE), "/text"
    ))
  if (http_type(rsp) != 'application/json') {
    stop("Failed to get first run info of: ", run)
  }
  jr <- content(rsp)
  ri <- data.frame(
      ModelName = jr$ModelName,
      ModelVersion = jr$ModelVersion,
      RunName = jr$Name,
      SubCount = jr$SubCount,
      RunStarted = jr$CreateDateTime,
      RunCompleted = jr$UpdateDateTime,
      RunDescription = "",
      RunNotes = ""
    )
  if (length(jr$Txt) > 0) {
    ri$RunDescription <- jr$Txt[[1]]$Descr
    ri$RunNotes <- jr$Txt[[1]]$Note
  }

  runInfo <- rbind(runInfo, ri)
}

# for each table do:
#   combine all run results and write it into some .xlsx file
#

lst <- list(
```

```
shts <- list(
    RunInfo = runInfo,
    TableInfo = tableInfo
  )

for (tbl in tn$TableNames)
{

  allCct <- NULL

  for (run in rn$RunNames)
  {
    cct <- read.csv(paste0(
      apiUrl, "model/", md, "/run/", URLencode(run, reserved = TRUE), "/table/", tbl, "/expr/csv"
      ))
    cct$RunName <- run

    allCct <- rbind(allCct, cct)
  }
  shts[[ tbl ]] <- allCct
}

write_xlsx(shts, paste0("output-tables-data.xlsx"))
```

Run model from Python: simple loop over model parameter

OpenM++ integration with Python

This example shows how Python can be used to automate modeling, using very general openM++ interfaces. These same interfaces can be used by platforms and applications other than Python with equivalent functionality.

Following Python script is running openM++ "NewCaseBased" test model with 16 subsamples using mortality hazard data:

```
mortalityData = [0.014 + i * 0.005 for i in range(20)]
```

As result Mortality Hazard increases about eight times in the range of [0.014, 0.109] and we can see eight time decrease of Duration of Life from initial 72 years down to 9 years.

How to run the script

Python example script is using openM++ web-service in order to run the model, modify parameters and read output values. OpenM++ web-service does not require any installation, just download latest release of openM++, unpack it into any directory, start `oms.exe` and run the script:

Windows:

```
cd C:\my-openmpp-release
bin\ompp_ui.bat
py ompp-python\life_vs_mortality.py
```

Linux / MacOS:

```
cd ~/my-openmpp-release
bin/oms
python3 ompp-python/life_vs_mortality.py
```

As result `oms` web-service will start to listen incoming requests on `http://localhost:4040` and Python script will do all actions using oms web-service API.

You may also need to install `mathplotlib` to display the chart and `requests` to communicate with web-service:

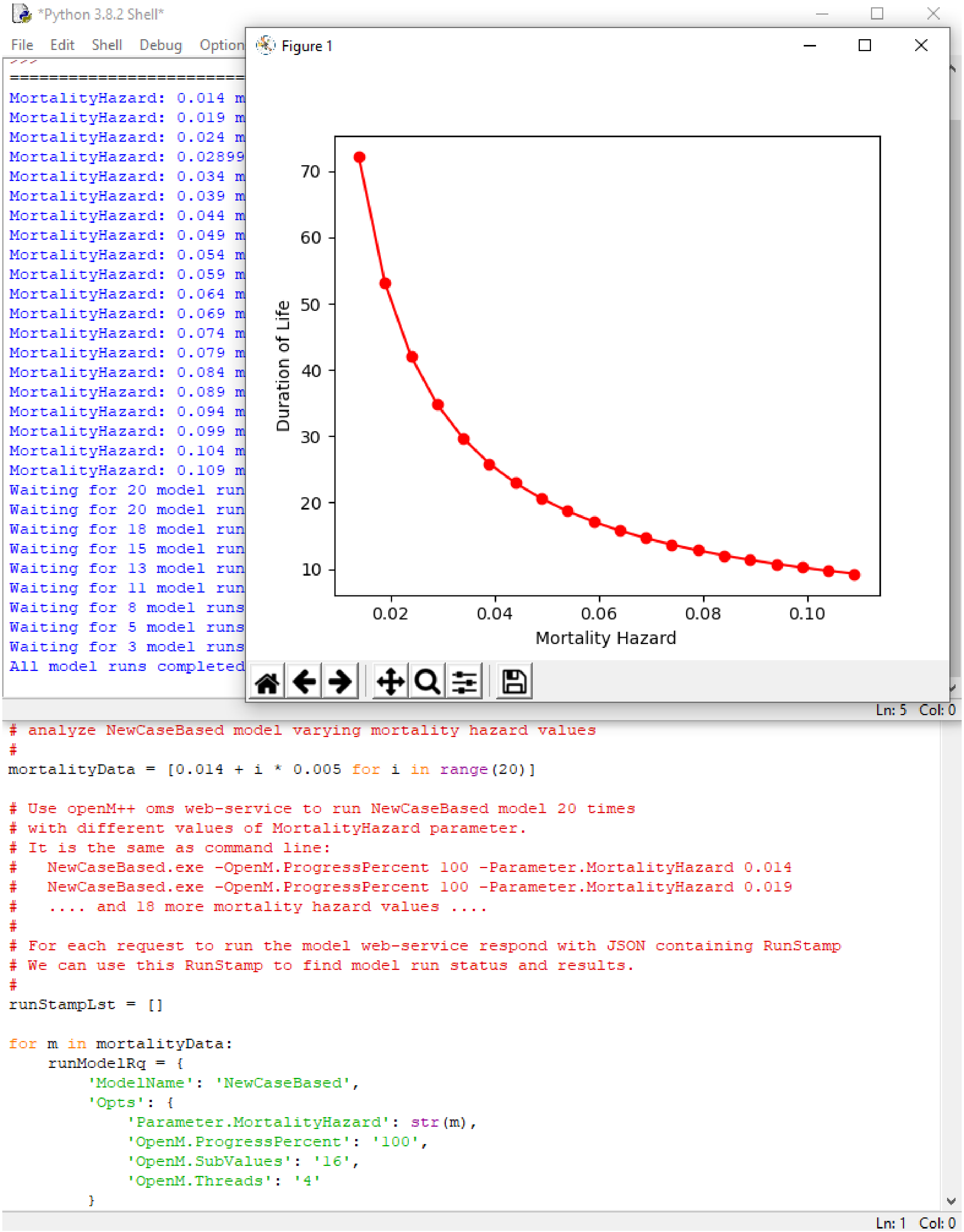
```
pip install -U matplotlib
pip install requests
```

Important:

This is an example script and error handling intentionally omitted. It is highly recommended to use `try ... except` in production code.

Important:

This is an example script and for simplicity it starts 20 instances of the model simultaneously. Obviously this can work only if model relatively simple. DO NOT USE this in production, please use modeling task instead.



Python script

```
#
# Python integration example using NewCaseBased model:
#   loop over MortalityHazard parameter
#   to analyze DurationOfLife output value

# Prerequisite:
#
# download openM++ release from https://github.com/openmpp/main/releases/latest
# unpack it into any directory
# start oms web-service:
#   Windows:
#     cd C:\my-openmpp-release
#     bin\ompp_ui.bat
#   Linux:
#     cd ~/my-openmpp-release
#     bin/oms
```

```
#
# Script below is using openM++ web-service "oms"
# to run the model, modify parameters and read output values.

# Important:
#   Script below starts 20 instances of the model simultaneously.
#   Obviously this can work only if model relatively simple.
#
# DO NOT USE this in production, please use modeling task instead.
#
# Also script below does not handle errors, please use try/except in production.

import time
import requests
import matplotlib.pyplot as plt

# analyze NewCaseBased model varying mortality hazard values
#
mortalityData = [0.014 + i * 0.005 for i in range(20)]

# Use openM++ oms web-service to run NewCaseBased model 20 times
# with different values of MortalityHazard parameter:
#
# NewCaseBased.exe -OpenM.ProgressPercent 100 -OpenM.SubValues 16 OpenM.Threads 4 -Parameter.MortalityHazard 0.014
# NewCaseBased.exe -OpenM.ProgressPercent 100 -OpenM.SubValues 16 OpenM.Threads 4 -Parameter.MortalityHazard 0.019
#   .... and 18 more mortality hazard values ....
#
# For each request to run the model web-service respond with JSON containing RunStamp
# We can use this RunStamp to find model run status and results.
#
runStampLst = []

for m in mortalityData:
    runModelRq = {
        'ModelName': 'NewCaseBased',
        'Opts': {
            'Parameter.MortalityHazard': str(m),
            'OpenM.ProgressPercent': '100',     # reduce amount of progress messages in the log file
            'OpenM.SubValues': '16',            # use 16 sub-values (sub-samples)
            'OpenM.Threads': '4'                # use 4 modeling threads
        }
    }
    #
    # submit request to web-service to run the model
    #
    rsp = requests.post('http://127.0.0.1:4040/api/run', json=runModelRq)
    rsp.raise_for_status()
    js = rsp.json()
    #
    runStamp = js['RunStamp']
    if runStamp is None or runStamp == '':
        raise Exception('Model fail to start, run stamp is empty')
    #
    runStampLst.append(runStamp)
    #
    print("MortalityHazard:", m, "model run stamp:", runStamp)

# wait until all model runs completed
#
n = len(runStampLst)
runDigestLst = ['' for i in range(n)]
done = [False for i in range(n)]

while n > 0:
    print("Waiting for", n, "model runs to be completed...")
    n = 0
    #
    for i in range(len(runStampLst)):
        if done[i]:
            continue    # run already completed
        #
        rsp = requests.get('http://127.0.0.1:4040/api/model/NewCaseBased/run/' + runStampLst[i] + '/status')
        rsp.raise_for_status()
        js = rsp.json()
        runDigestLst[i], status = js['RunDigest'], js['Status']
        #
        if runDigestLst[i] is None or runDigestLst[i] == '' or \
            status is None or status == '' or \
            status in 'i' 'p':  # i = run not started yet, p = run in progress
            #
            n += 1
            continue
        #
        if status == 's':       # success
            done[i] = True
            continue
        #
        raise Exception("Model run failed, run stamp:", runStampLst[i], "status:", status)
```

```
        raise Exception('Model run failed, run stamp: ', runStampLst[i], 'status: ', status)
        #
    #
    if n > 0:
        time.sleep(1)

# all model runs completed successfully
print("All model runs completed, retrive output values...")

# for each run get output value
#   average duration of life: DurationOfLife.Expr3
#
lifeDurationData = []

for runDigest in runDigestLst:
    rsp = requests.get('http://127.0.0.1:4040/api/model/NewCaseBased/run/' + runDigest + '/table/DurationOfLife/expr')
    rsp.raise_for_status()
    js = rsp.json()
    lifeDurationData.append(js[3]['Value'])

# display the results
#
plt.plot(mortalityData, lifeDurationData, 'ro', ls='-')
plt.xlabel('Mortality Hazard')
plt.ylabel('Duration of Life')
plt.show()
```

Run RiskPaths model from Python: advanced parameters scaling

OpenM++ integration with Python: using RiskPaths model

This example shows how Python can be used to automate modeling, using very general openM++ interfaces. These same interfaces can be used by platforms and applications other than Python with equivalent functionality.

Following Python script is running "RiskPaths" model to analyze childlessness by varying two parameters:

- Age baseline for first union formation
- Relative risks of union status on first pregnancy by following scale factor:

  ```
  scaleStep = 0.02
  scaleValues = [0.44 + i * scaleStep for i in range(1 + round((1.00 - 0.44) / scaleStep))]
  ```

  Please keep in mind, scaling above result in 841 runs of RiskPaths model and task may take long time to be completed. If you want to get results faster scale values by 0.08 instead of 0.02.

How to run the script

Python example script is using openM++ web-service in order to run the model, modify parameters and read output values. OpenM++ web-service does not require any installation, just download latest release of openM++, unpack it into any directory, start `oms.exe` and run the script:

Windows:

```
cd C:\my-openmpp-release
bin\oms
py ompp-python\riskpaths_childlessness.py
```

Linux / MacOS:

```
cd ~/my-openmpp-release
bin/oms
python3 ompp-python/riskpaths_childlessness.py
```

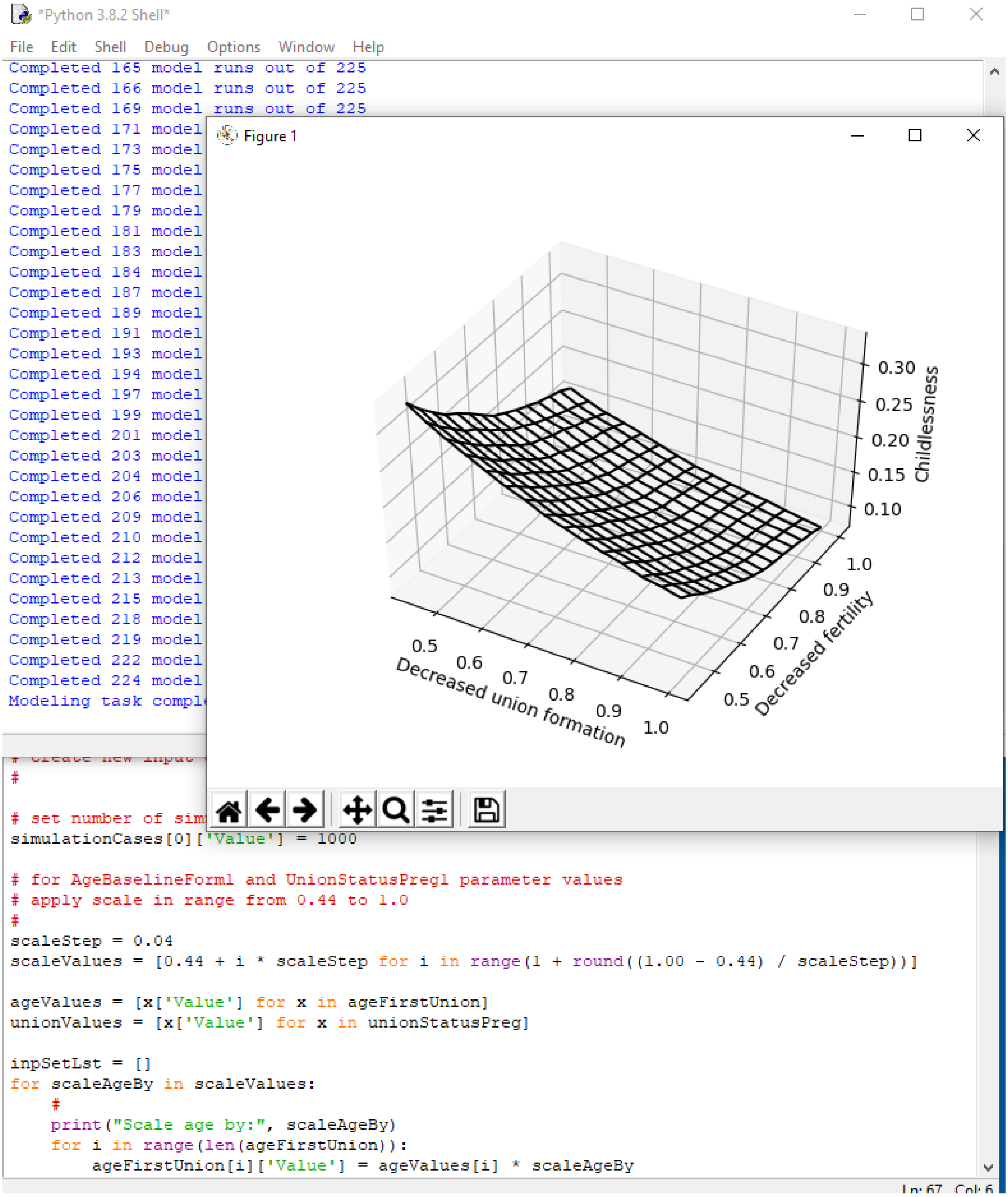
As result `oms` web-service will start to listen incoming requests on `http://localhost:4040` and Python script will do all actions using oms web-service API.

You may also need to install `mathplotlib` to display the chart and `requests` to communicate with web-service:

```
pip install -U matplotlib
pip install requests
```

Important:

This is an example script and error handling intentionally omitted. It is highly recommended to use `try ... except` in production code.



Python script

```
#
# Python integration example using RiskPaths model
#   to analyze contribution of delayed union formations
#   versus decreased fertility on childlessness
#
# Input parameters:
#   AgeBaselineForm1: age baseline for first union formation
#   UnionStatusPreg1: relative risks of union status on first pregnancy
# Output value:
#   T05_CohortFertility: Cohort fertility, expression 1
#

# Prerequisite:
#
# download openM++ release from https://github.com/openmpp/main/releases/latest
# unpack it into any directory
# start oms web-service:
#   Windows:
#     cd C:\my-openmpp-release
#     bin\ompp_ui.bat
#   Linux:
```

```
#   Linux:
#     cd ~/my-openmpp-release
#     bin/oms
#
# Script below is using openM++ web-service "oms"
# to run the model, modify parameters and read output values.

# Important:
#   Script below does not handle errors, please use try/except in production.

import time
import requests
import numpy as np
import matplotlib.pyplot as plt

# get default values for AgeBaselineForm1, UnionStatusPreg1 and SimulationCases parameters
# by reading it from first model run results
# assuming first run of the model done with default set of parameters
#
rsp = requests.get('http://127.0.0.1:4040/api/model/RiskPaths/run/status/first')
rsp.raise_for_status()
firstRunStatus = rsp.json()
firstRunDigest = rsp.json()['RunDigest']

rsp = requests.get('http://127.0.0.1:4040/api/model/RiskPaths/run/' + firstRunDigest + '/parameter/AgeBaselineForm1/value/start/0/count/0')
rsp.raise_for_status()
ageFirstUnion = rsp.json()

rsp = requests.get('http://127.0.0.1:4040/api/model/RiskPaths/run/' + firstRunDigest + '/parameter/UnionStatusPreg1/value/start/0/count/0')
rsp.raise_for_status()
unionStatusPreg = rsp.json()

rsp = requests.get('http://127.0.0.1:4040/api/model/RiskPaths/run/' + firstRunDigest + '/parameter/SimulationCases/value/start/0/count/0')
rsp.raise_for_status()
simulationCases = rsp.json()

# create new input data for our modelling task
#

# set number of simulation cases
simulationCases[0]['Value'] = 1000

# for AgeBaselineForm1 and UnionStatusPreg1 parameter values
# apply scale in range from 0.44 to 1.0
#
scaleStep = 0.02
scaleValues = [0.44 + i * scaleStep for i in range(1 + round((1.00 - 0.44) / scaleStep))]

ageValues = [x['Value'] for x in ageFirstUnion]
unionValues = [x['Value'] for x in unionStatusPreg]

inpSetLst = []
for scaleAgeBy in scaleValues:
    #
    print("Scale age by:", scaleAgeBy)
    for i in range(len(ageFirstUnion)):
        ageFirstUnion[i]['Value'] = ageValues[i] * scaleAgeBy

    for scaleUnionBy in scaleValues:
        #
        # scale first two values of unionStatusPreg vector
        unionStatusPreg[0]['Value'] = unionValues[0] * scaleUnionBy
        unionStatusPreg[1]['Value'] = unionValues[1] * scaleUnionBy
        #
        # create new set of input parameters
        # automatically generate unique names for each input set
        #
        inpSetRq = {
            'ModelName': 'RiskPaths',
            'Name': '',
            'BaseRunDigest': firstRunDigest,
            'IsReadonly': True,
            'Txt': [{
                'LangCode': 'EN',
                'Descr': 'Scale age: ' + str(scaleAgeBy) + ' union status: ' + str(scaleUnionBy)
            }],
            'Param': [
                {
                    'Name': 'AgeBaselineForm1',
                    'SubCount': 1,
                    'Value': ageFirstUnion,
                    'Txt': [{'LangCode': 'EN', 'Note': 'Age values scale by: ' + str(scaleAgeBy)}]
                },
                {
                    'Name': 'UnionStatusPreg1',
                    'SubCount': 1,
                    'Value': unionStatusPreg,
                    'Txt': [{'LangCode': 'EN', 'Note': 'Union Status values scale by: ' + str(scaleUnionBy)}]
                },
```

```
            }
        ],
    }
    #
    # create new input set of model parameters
    # automatically generate unique name for that input set
    #
    rsp = requests.put('http://127.0.0.1:4040/api/workset-create', json=inpSetRq)
    rsp.raise_for_status()
    js = rsp.json()
    #
    inpSetName = js['Name']
    if inpSetName is None or inpSetName == '':
        raise Exception("Fail to create input set, scales:", scaleAgeBy, scaleUnionBy)
    #
    inpSetLst.append(inpSetName)

# create modeling task from all input sets
# automatically generate unique name for the task
#
inpLen = len(inpSetLst)
print("Create task from", inpLen, "input sets of parameters")

taskRq = {
    'ModelName': 'RiskPaths',
    'Name': '',
    'Set': inpSetLst,
    'Txt': [{
        'LangCode': 'EN',
        'Descr': 'Task to run RiskPaths ' + str(inpLen) + ' times',
        'Note': 'Task scales AgeBaselineForm1 and UnionStatusPreg1 parameters from 0.44 to 1.00 with step ' + str(scaleStep)
    }]
}
rsp = requests.put('http://127.0.0.1:4040/api/task-new', json=taskRq)
rsp.raise_for_status()
js = rsp.json()

taskName = js['Name']
if taskName is None or taskName == '':
    raise Exception("Error at create modeling task")

#
# submit request to web-service to run RiskPaths with modeling task
#
runModelRq = {
    'ModelName': 'RiskPaths',
    'Opts': {
        'OpenM.TaskName': taskName,
        'OpenM.ProgressPercent': '100'
    }
}
rsp = requests.post('http://127.0.0.1:4040/api/run', json=runModelRq)
rsp.raise_for_status()
js = rsp.json()
#
taskRunStamp = js['RunStamp']
if taskRunStamp is None or taskRunStamp == '':
    raise Exception('Model failed to start, task run stamp is empty')

print("Starting modeling task:", taskName)

# wait until modeling task completed
# and report the progress
#
# task status returned by web-service can be one of:
# i=initial p=in progress w=waiting s=success x=exit e=error(failed)
#
taskStatus = ''

while taskStatus in '' 'i' 'p' 'w':
    #
    time.sleep(1)
    #
    rsp = requests.get('http://127.0.0.1:4040/api/model/RiskPaths/task/' + taskName + '/run-status/run/' + taskRunStamp)
    rsp.raise_for_status()
    js = rsp.json()
    taskStatus = js['Status']
    #
    # if model not started to run the task yet check again after short sleep
    #
    if taskStatus in '' 'i':
        #
        print("Waiting for modeling task to start...")
        continue
    #
    # if task completed successfully then get pairs of {model run, inpur set name}
    #
    if taskStatus == 's':
```

```
        rsp = requests.get('http://127.0.0.1:4040/api/model/RiskPaths/task/' + taskName + '/runs')
        rsp.raise_for_status()
        js = rsp.json()
        taskRuns = js['TaskRun'][0]['TaskRunSet']   # use index=0 because this is first run of our task
        break
    #
    # if task still in progress then count completed model runs
    #
    if taskStatus in 'i' 'p' 'w':
        rsp = requests.get('http://127.0.0.1:4040/api/model/RiskPaths/run/' + taskRunStamp + '/status/list')
        rsp.raise_for_status()
        trs = rsp.json()
        #
        n = 0
        for r in trs:
            if r['Status'] == 's': n += 1
        #
        print("Completed", n, "model runs out of", inpLen)
        continue
    #
    # any other task run status considered as failure
    #
    raise Exception("Model run failed, task run stamp:", taskRunStamp, "status:", taskStatus)
    #

print("Modeling task completed, retriving results...")

# for each age and union status retrive output:
#   childlessness value: T05_CohortFertility.Expr1
#
# organize results into 2-dimensional array to plot 3d chart
#
childlessnessVals = np.zeros((len(scaleValues), len(scaleValues)))
runIdx = 0

for ageIdx in range(len(scaleValues)):
    for unionIdx in range(len(scaleValues)):
        #
        runDigest = taskRuns[runIdx]['Run']['RunDigest']
        #
        rsp = requests.get('http://127.0.0.1:4040/api/model/RiskPaths/run/' + runDigest + '/table/T05_CohortFertility/expr')
        rsp.raise_for_status()
        js = rsp.json()
        #
        childlessnessVals[ageIdx][unionIdx] = js[1]['Value']
        runIdx += 1

# display the results
#
ageVals, unionVals = np.meshgrid(scaleValues, scaleValues)

fig = plt.figure()
ax = plt.axes(projection='3d')
ax.plot_wireframe(ageVals, unionVals, childlessnessVals, color='black')
ax.set_xlabel('Decreased union formation')
ax.set_ylabel('Decreased fertility')
ax.set_zlabel('Childlessness')
ax.view_init(elev=45)
plt.show()
```

Run model from R: simple loop over model parameter

OpenM++ integration with R

It is a convenient to use GNU R to prepare model parameters and analyze output values. There are two different R APIs which we can use for openM++ models:

- openMpp package: simple and convenient specially for desktop users, upstream and downstream analysis;
- `oms` JSON web-service API: preferable choice to run models on computational clusters and in cloud.

Below is a simple loop example to run NewCaseBased model on desktop using openMpp R package. There is similar example how to run model in cloud and save results in CSV file using `oms` JSON web-service.

OpenM++ provides R package `openMpp` to simplify access to openM++ database for R developers. To find out more about openMpp R package please check:

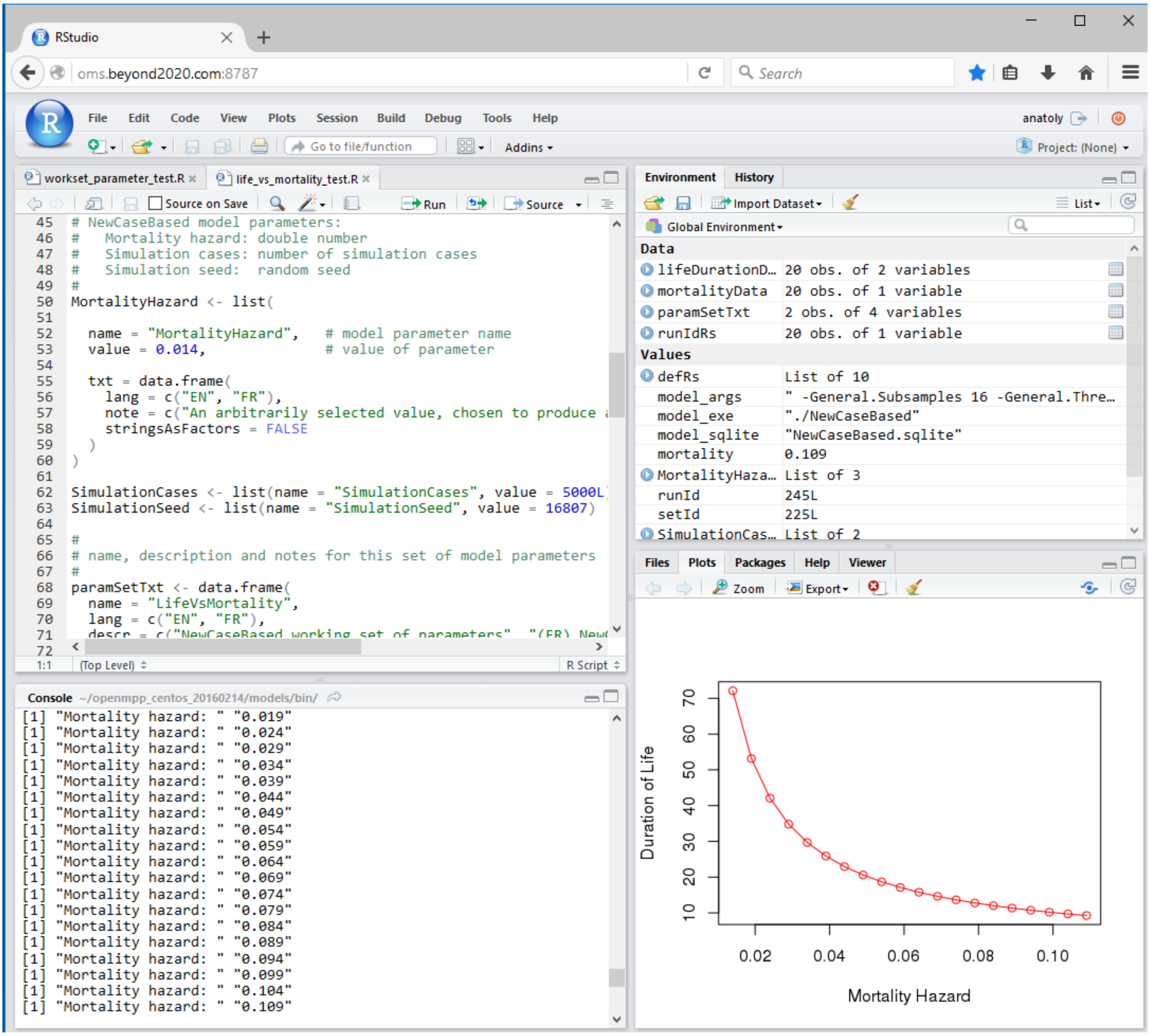
- openMpp package documentation
- installation ReadMe and source code

As it is today openMpp package tested only with R up to version 3.6.3 and R version 4 has not been tested yet.

Following R example is running openM++ "NewCaseBased" test model with 16 subsamples using mortality hazard data:

```
mortalityData <- data.frame(
  value = seq(from = 0.014, by = 0.005, length.out = 20)
)
```

As result Mortality Hazard increases about eight times in the range of [0.014, 0.109] and we can see eight time decrease of Duration of Life from initial 72 years down to 9 years.



R script

```
# use openMpp library for openM++ database access
library(DBI)
library("openMpp")
library("RSQLite")

#
# R integration example using NewCaseBased model
#   loop over MortalityHazard parameter
#   to analyze DurationOfLife
#

##################################################################
# To run this example please uncomment and/or change values below
# to match your hardware and file system environment:
#
# model_exe    <- path to the model executable, i.e.: "./NewCaseBased" or "NewCaseBased.exe"
# model_sqlite <- path to the model.sqlite database:  "NewCaseBased.sqlite"
# model_args   <- optional arguments to control model run, for example:
#       -OpenM.SubValues 16 <- number of simation members
#       -OpenM.Threads 4    <- number of computatinal threads
#
### For running on a local machine using the working directory in R
#
# For the following values to work, you must first set the R Working directory
# to the directory containing the NewCaseBased executable and the SQLite database.
# In RStudio Session > Set Working Directory > Choose Directory,
# then navigate to location, e.g.: /OM_ROOT/models/NewCaseBased/ompp/bin
# Alternatively, one may use setwd(), e.g.: setwd("/OM_ROOT/models/NewCaseBased/ompp/bin")
#
```

```r
model_exe = "./NewCaseBased"
model_sqlite = "NewCaseBased.sqlite"
model_args = " -OpenM.SubValues 16 -OpenM.Threads 4"
# model_args = " "  # default: 1 simulation member and 1 thread
#
### For running on a local machine using explicit paths
#
# model_exe = "/path/to/executable/model/NewCaseBased"
# model_sqlite = "/path/to/SQLite/database/NewCaseBased.sqlite"
#
### For running on cluster (change to match your cluster)
#
# model_exe = "mpiexec"
# model_sqlite = "/mirror/NewCaseBased.sqlite"
# model_args = "-n 8 /mirror/NewCaseBased -OpenM.SubValues 16 -OpenM.Threads 2"
##################################################################

#
# NewCaseBased model parameters:
#   Mortality hazard: double number
#   Simulation cases: number of simulation cases
#   Simulation seed:  random seed
#
MortalityHazard <- list(

  name = "MortalityHazard",   # model parameter name
  value = 0.014,              # value of parameter

  txt = data.frame(
    lang = c("EN", "FR"),
    note = c("An arbitrarily selected value, chosen to produce a life expectancy of about 70 years", NA),
    stringsAsFactors = FALSE
  )
)

SimulationCases <- list(name = "SimulationCases", value = 5000L)
SimulationSeed <- list(name = "SimulationSeed", value = 16807)

#
# name, description and notes for this set of model parameters
#
inputSet <- data.frame(
  name = "LifeVsMortality",
  lang = c("EN", "FR"),
  descr = c("NewCaseBased working set of parameters", "(FR) NewCaseBased working set of parameters"),
  note = c(NA, NA),
  stringsAsFactors = FALSE
)

#
# connect to database and find NewCaseBased model
#
theDb <- dbConnect(RSQLite::SQLite(), model_sqlite, synchronous = "full")
invisible(dbGetQuery(theDb, "PRAGMA busy_timeout = 86400"))  # recommended

defRs <- getModel(theDb, "NewCaseBased")      # find NewCaseBased model in database

# create new working set of model parameters based on existing model run results
#
firstRunId <- getFirstRunId(theDb, defRs)

setId <- createWorksetBasedOnRun(
  theDb, defRs, firstRunId, inputSet,
  MortalityHazard, SimulationCases, SimulationSeed
  )
if (setId <= 0L) stop("workset creation failed")

setReadonlyWorkset(theDb, defRs, TRUE, setId)  # workset must be read-only to run the model

#
# analyze NewCaseBased model varying mortality hazard values
#
mortalityData <- data.frame(
  value = seq(from = 0.014, by = 0.005, length.out = 20)
)

for (mortality in mortalityData$value)
{
  print(c("Mortality hazard: ", mortality))

  system2(
    model_exe,
    paste(
      model_args,
      " -Parameter.MortalityHazard ", mortality,
      " -OpenM.SetId ", setId,
      " -OpenM.LogToConsole false",
      " -OpenM.LogToFile true",
```

```
      " -OpenM.ProgressPercent 100",
      sep = ""
    )
  )
}

#
# read final results from database
#   average duration of life: DurationOfLife.Expr3
#
runIdRs <- getWorksetRunIds(theDb, setId)   # get result id's

lifeDurationData <- NULL
for (runId in runIdRs$run_id)
{
  lifeDurationData <- rbind(
    lifeDurationData,
    selectRunOutputValue(theDb, defRs, runId, "DurationOfLife", "Expr3")
  )
}

dbDisconnect(theDb)   # close database connection

#
# display the results
#
plot(
  mortalityData$value,
  lifeDurationData$expr_value,
  type = "o",
  xlab = "Mortality Hazard",
  ylab = "Duration of Life",
  col = "red"
)
```

Run model from R: simple loop in cloud

OpenM++ integration with R: run model and save results in CSV file

It is a convenient to use GNU R to prepare model parameters and analyze output values. There are two different R APIs which we can use for openM++ models:

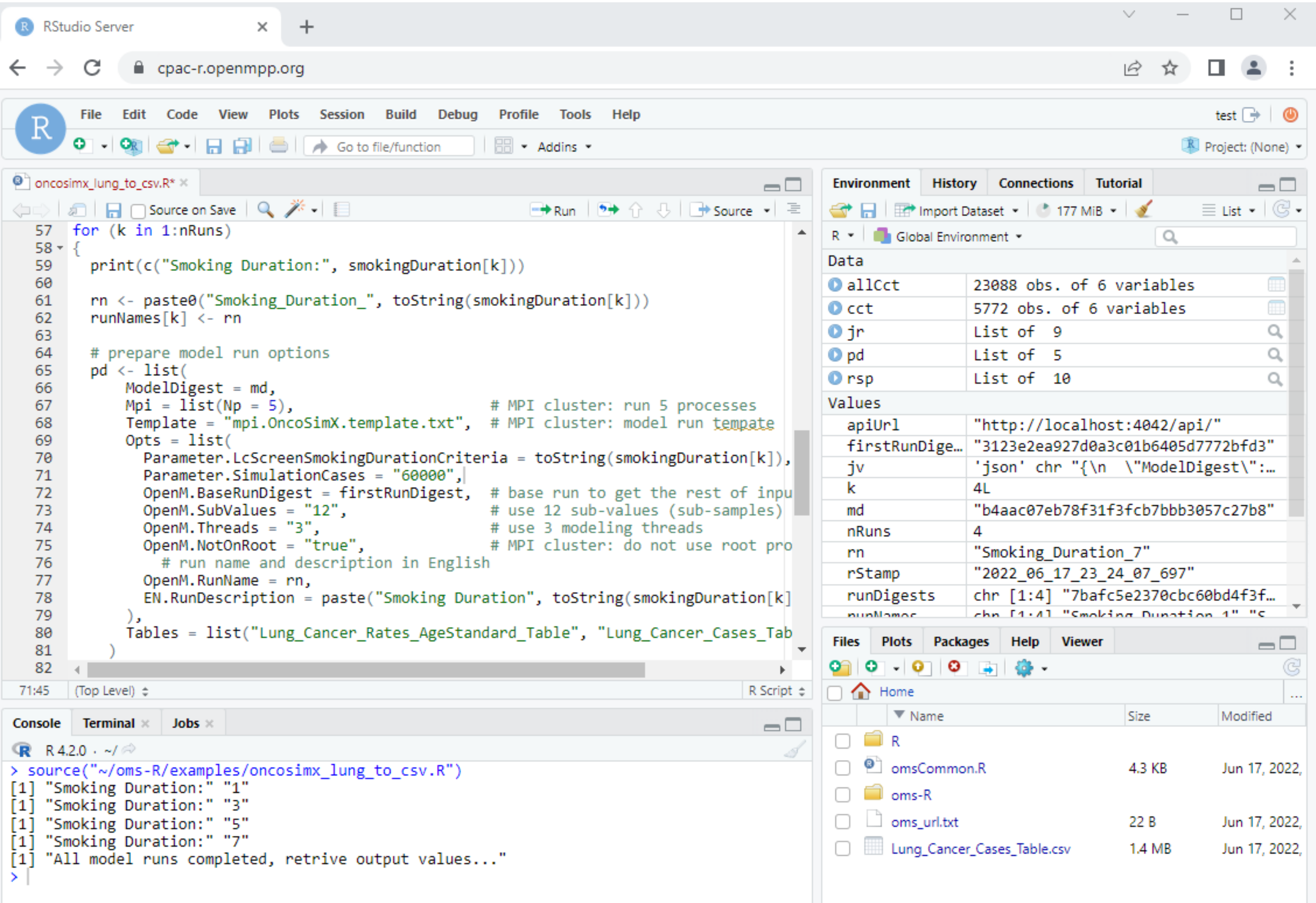
- openMpp package: simple and convenient specially for desktop users, upstream and downstream analysis;
- `oms` JSON web-service API: preferable choice to run models on computational clusters and in cloud.

Below is an example how to run model in cloud and save results in CSV file using `oms` JSON web-service. There is a similar example how to run model on desktop and do simple loop over parameter using openMpp R package.

Following R example is running very complex OncoSimX-lung model to change only `LcScreenSmokingDurationCriteria` parameter:

```
smokingDuration <- seq(from = 1, by = 2, length.out = 4)
```

To reduce model run time we are calculating only 2 output tables: `Lung_Cancer_Rates_AgeStandard_Table` and `Lung_Cancer_Cases_Table` and also using only 6000 simulation cases. Also we do merge `Lung_Cancer_Cases_Table` rows from all model runs and saving it into `Lung_Cancer_Cases_Table.csv`



R script

```
#
# Use R to run OncoSimX-lung version 3.5.0.90
#   loop over LcScreenSmokingDurationCriteria parameter
#   to output tables: Lung_Cancer_Rates_AgeStandard_Table and Lung_Cancer_Cases_Table
#

# If jsonlite or httr is not installed then do:
#   install.packages("jsonlite")
#   install.packages("httr")
#
library("jsonlite")
library("httr")

# Include openM++ helper functions from your $HOME directory
```

```r
# include openM++ helper functions from your $HOME directory
# on Windows HOME directory is: "C:\Users\User Name Here\Documents"
#
# if you don't have omsCommon.R then download it from https://github.com/openmpp/R/oms-R
# if you have omsCommon.R in some other location then update path below
#
source("~/omsCommon.R")

#
# Model digest of OncoSimX-lung version 3.5.0.90: "b4aac07eb78f31f3fcb7bbb3057c27b8"
#
md <- "b4aac07eb78f31f3fcb7bbb3057c27b8"

# oms web-service URL from file: ~/oms_url.txt
#
apiUrl <- getOmsApiUrl()

# Find first model run to use it as our base run
#
rsp <- GET(paste0(
    apiUrl, "model/", md, "/run/status/first"
  ))
if (http_type(rsp) != 'application/json') {
  stop("Failed to get first run status")
}
jr <- content(rsp)
firstRunDigest <- jr$RunDigest

# Use openM++ oms web-service to run the model 4 times with 6000 simulation cases
# and different values of LcScreenSmokingDurationCriteria parameter:
#
# OncoSimX-lung_mpi -Parameter.SimulationCases 6000 -Parameter.LcScreenSmokingDurationCriteria 1
# OncoSimX-lung_mpi -Parameter.SimulationCases 6000 -Parameter.LcScreenSmokingDurationCriteria 3
#   .... and 2 more Smoking Duration values ....
#
# Use back-end cluster to run the model with 12 sub-values on 4 servers and 3 theards
#
# It is MPI model on small computational cluster of 4 servers
# running 5 OncoSimX-lung_mpi instancces: "root" leader process and 4 computational processes
# each computational process using modelling 3 threads
# root process does only database operations and coordinate child workoload.
#
nRuns <- 4
smokingDuration <- seq(from = 1, by = 2, length.out = nRuns)

runDigests <- rep('', nRuns)  # model run digests, unique
runNames <- rep('', nRuns)    # model run names, may be not unique

for (k in 1:nRuns)
{
  print(c("Smoking Duration:", smokingDuration[k]))

  rn <- paste0("Smoking_Duration_", toString(smokingDuration[k]))
  runNames[k] <- rn

  # prepare model run options
  pd <- list(
      ModelDigest = md,
      Mpi = list(Np = 5),                    # MPI cluster: run 5 processes
      Template = "mpi.OncoSimX.template.txt",  # MPI cluster: model run tempate
      Opts = list(
        Parameter.LcScreenSmokingDurationCriteria = toString(smokingDuration[k]),
        Parameter.SimulationCases = "6000",
        OpenM.BaseRunDigest = firstRunDigest,  # base run to get the rest of input parameters
        OpenM.SubValues = "12",               # use 12 sub-values (sub-samples)
        OpenM.Threads = "3",                  # use 3 modeling threads
        OpenM.NotOnRoot = "true",             # MPI cluster: do not use root process for modelling
          # run name and description in English
        OpenM.RunName = rn,
        EN.RunDescription = paste("Smoking Duration", toString(smokingDuration[k]), "years")
      ),
      Tables = list("Lung_Cancer_Rates_AgeStandard_Table", "Lung_Cancer_Cases_Table")
    )
  jv <- toJSON(pd, pretty = TRUE, auto_unbox = TRUE)

  # submit request to web-service to run the model
  rsp <- POST(paste0(
        apiUrl, "run"
      ),
      body = jv,
      content_type_json()
    )
  if (http_type(rsp) != 'application/json') {
    stop("Failed to run the model")
  }
  jr <- content(rsp)
  rStamp <- jr$RunStamp  # model run stamp
```

```
  # wait until model run completed
  runDigests[k] <- waitForRunCompleted(rStamp, apiUrl, md)
}

# combine all run results into Lung_Cancer_Cases_Table.csv
#
print("All model runs completed, retrive output values...")

allCct <- NULL

for (k in 1:nRuns)
{
  cct <- read.csv(paste0(
    apiUrl, "model/", md, "/run/", runDigests[k], "/table/Lung_Cancer_Cases_Table/expr/csv"
    ))
  cct$RunName <- runNames[k]

  allCct <- rbind(allCct, cct)
}

write.csv(allCct, "Lung_Cancer_Cases_Table.csv", row.names = FALSE)
```

Run RiskPaths model from R: advanced parameters scaling

OpenM++ integration with R: using RiskPaths model

It is a convenient to use GNU R to prepare model parameters and analyze output values. There are two different R APIs which we can use for openM++ models:

- openMpp package: simple and convenient specially for desktop users, upstream and downstream analysis;
- `oms` JSON web-service API: preferable choice to run models on computational clusters and in cloud.

Below is an example how to do advanced parameters analysis and run RiskPaths model on desktop using openMpp R package. There is an identical example how to run RiskPaths on cloud computational grid using of `oms` JSON web-service.

OpenM++ provides R package `openMpp` to simplify access to openM++ database for R developers. To find out more about openMpp R package please check:

- openMpp package documentation
- installation ReadMe and source code

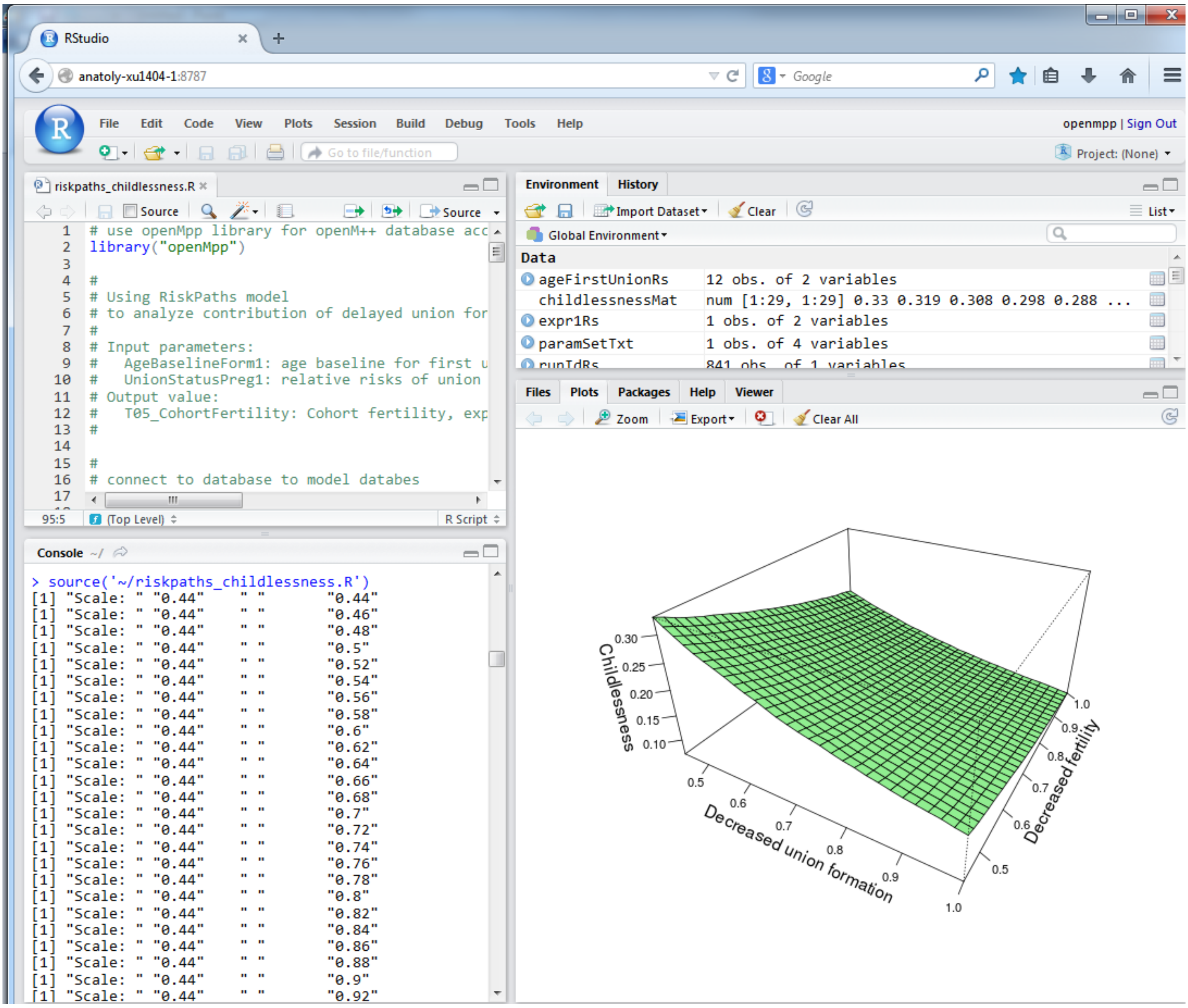
As it is today openMpp package tested only with R up to version 3.6.3 and R version 4 has not been tested yet.

Following R example is running "RiskPaths" model to analyze childlessness by varying two parameters:

- Age baseline for first union formation
- Relative risks of union status on first pregnancy by following scale factor:

  ```
  scaleValues <- seq(from = 0.44, to = 1.00, by = 0.02)
  ```

  Please keep in mind, scaling above result in 841 runs of RiskPaths model and task may take long time to be completed. If you want to get results faster scale values by 0.08 instead of 0.02.



R script

```
# use openMpp library for openM++ database access
library(DBI)
library("openMpp")
library("RSQLite")

#
# Using RiskPaths model
#   to analyze contribution of delayed union formations
#   versus decreased fertility on childlessness
#
# Input parameters:
#   AgeBaselineForm1: age baseline for first union formation
#   UnionStatusPreg1: relative risks of union status on first pregnancy
# Output value:
#   T05_CohortFertility: Cohort fertility, expression 1
#

######################################################################
# To run this example please uncomment and/or change values below
# to match your hardware and file system environment:
#
# model_exe    <- path to the model executable, i.e.: "./RiskPaths" or "RiskPaths.exe"
# model_sqlite <- path to the model.sqlite database:  "RiskPaths.sqlite"
# model_args   <- optional arguments to control model run, for example:
#       -OpenM.SubValues 8 <- number of simation members
#       -OpenM.Threads 4   <- number of computatinal threads
#
### For running on a local machine using the working directory in R
#
# For the following values to work, you must first set the R Working directory
# to the directory containing the RiskPaths executable and the SQLite database.
# In RStudio Session > Set Working Directory > Choose Directory,
# then navigate to location, e.g.: /OM_ROOT/models/RiskPaths/ompp/bin
```

```r
# then navigate to location, e.g.: /OM_ROOT/models/RiskPaths/ompp/bin
# Alternatively, one may use setwd(), e.g.: setwd("/OM_ROOT/models/RiskPaths/ompp/bin")
#
model_exe = "./RiskPaths"
model_sqlite = "RiskPaths.sqlite"
model_args = " "  # default: 1 simulation member and 1 thread
# model_args = " -OpenM.SubValues 8 -OpenM.Threads 4"
#
### For running on a local machine using explicit paths
#
# model_exe = "/path/to/executable/model/RiskPaths"
# model_sqlite = "/path/to/SQLite/database/RiskPaths.sqlite"
#
### For running on cluster (change to match your cluster)
#
# model_exe = "mpiexec"
# model_sqlite = "/mirror/RiskPaths.sqlite"
# model_args = "-n 8 /mirror/RiskPaths -OpenM.SubValues 16 -OpenM.Threads 2"
##################################################################

#
# connect to database to model databes
#
theDb <- dbConnect(RSQLite::SQLite(), model_sqlite, synchronous = "full")
invisible(dbGetQuery(theDb, "PRAGMA busy_timeout = 86400"))   # recommended

# find RiskPaths model in database and get model dictionaries ("modelDic", "typeDic", etc...)
defRs <- getModel(theDb, "RiskPaths")

#
# create a copy of default model parameters
#
baseRunId <- getFirstRunId(theDb, defRs)
if (baseRunId <= 0)
  stop("no run results found for the model ", defRs$modelDic$model_name, " ", defRs$modelDic$model_digest)

#
# get default values for AgeBaselineForm1 and UnionStatusPreg1 parameters
# by reading it from first model run results
# assuming first run of the model done with default set of parameters
#
ageFirstUnionRs   <- selectRunParameter(theDb, defRs, baseRunId, "AgeBaselineForm1")
unionStatusPregRs <- selectRunParameter(theDb, defRs, baseRunId, "UnionStatusPreg1")

#
# create modeling task with
# all input parameters same as model default except of
# AgeBaselineForm1, UnionStatusPreg1 and SimulationCases parameters
#
casesParam <- list(name = "SimulationCases", value = 1000L) # number of simulation cases

taskTxt <- data.frame(      # name (auto generated), description and notes for the task
  name  = NA,
  lang  = "EN",
  descr = "Analyzing childlessness",
  note  = NA,
  stringsAsFactors = FALSE
)

taskId <- createTask(theDb, defRs, taskTxt)
if (taskId <= 0L) stop("task creation failed: ", defRs$modelDic$model_name, " ", defRs$modelDic$model_digest)

# parameters scale
#
# scaleValues <- seq(from = 0.50, to = 1.00, by = 0.50) # tiny set of runs for quick test
#
scaleValues <- seq(from = 0.44, to = 1.00, by = 0.02)

UnionStatusMultipler = rep(1, length(unionStatusPregRs$param_value)) # vector of 1's

for (scAgeBy in scaleValues)
{
  print(c("Scale age: ", scAgeBy))

  for (scUnionBy in scaleValues)
  {
    ageParam <- list(name = "AgeBaselineForm1", value = ageFirstUnionRs$param_value * scAgeBy)

    UnionStatusMultipler[1:2] = scUnionBy  # scale first two values of parameter vector
    unionParam <- list(name = "UnionStatusPreg1", value = unionStatusPregRs$param_value *  UnionStatusMultipler )

    # Append new working set of parameters into the task. A corresponding setId is generated.
    setId <- createWorksetBasedOnRun(theDb, defRs, baseRunId, NA, ageParam, unionParam, casesParam)
    setReadonlyWorkset(theDb, defRs, TRUE, setId)

    taskId <- updateTask(theDb, defRs, taskId, setIds = setId)
  }
}
```

```r
#
# run the model on cluster or local desktop
# consult your cluster admin on how to use computational grid
print(paste("Run the model:", model_exe, "...please wait..."))

system2(
  model_exe,
  paste(
    model_args,
    " -OpenM.TaskId ", taskId,
    " -OpenM.LogToConsole false",
    " -OpenM.LogToFile true",
    " -OpenM.ProgressPercent 100",
    sep = ""
  )
)

#
# read results of task run from database
#   cohort fertility: T05_CohortFertility.Expr1
#
taskRunId <- getTaskLastRunId(theDb, taskId)  # most recent task run id
taskRunRs <- selectTaskRun(theDb, taskRunId)  # get result id's
#
# taskRunId
# [1] 111
# taskRunRs$taskRunSet  # Content for "tiny set of runs"
#   task_run_id run_id set_id task_id
# 1         108    109    104     103
# 2         108    110    105     103
# 3         108    111    106     103
# 4         108    112    107     103
# Main scenario task_id 103 comes with 4 sets of parameters  set_id 104, 105, 106, 107  (e.g. PSA)
# The main scenario/task was run (task_run_id 108) which spins out 4 runs run_id 109, 110, 111, 112

scaleLen <- length(scaleValues)
childlessnessMat <- matrix(data = NA, nrow = scaleLen, ncol = scaleLen, byrow = TRUE)

runPos <- 1
for (k in 1:scaleLen)
{
  for (j in 1:scaleLen)
  {
    # cohort fertility: T05_CohortFertility.Expr1
    expr1Rs <- selectRunOutputValue(theDb, defRs, taskRunRs$taskRunSet$run_id[runPos], "T05_CohortFertility", "Expr1")
    childlessnessMat[k, j] = expr1Rs$expr_value
    runPos <- runPos + 1
  }
}

dbDisconnect(theDb)   # close database connection

#
# display the results
#
persp(
  x = scaleValues,
  y = scaleValues,
  z = childlessnessMat,
  xlab = "Decreased union formation",
  ylab = "Decreased fertility",
  zlab = "Childlessness",
  theta = 30, phi = 30, expand = 0.5, ticktype = "detailed",
  col = "lightgreen",
  cex.axis = 0.7
)
```

Run RiskPaths model from R: advanced run in cloud

OpenM++ integration with R: run RiskPaths model on cloud grid

It is a convenient to use GNU R to prepare model parameters and analyze output values. There are two different R APIs which we can use for openM++ models:

- openMpp package: simple and convenient specially for desktop users, upstream and downstream analysis;
- `oms` JSON web-service API: preferable choice to run models on computational clusters and in cloud.

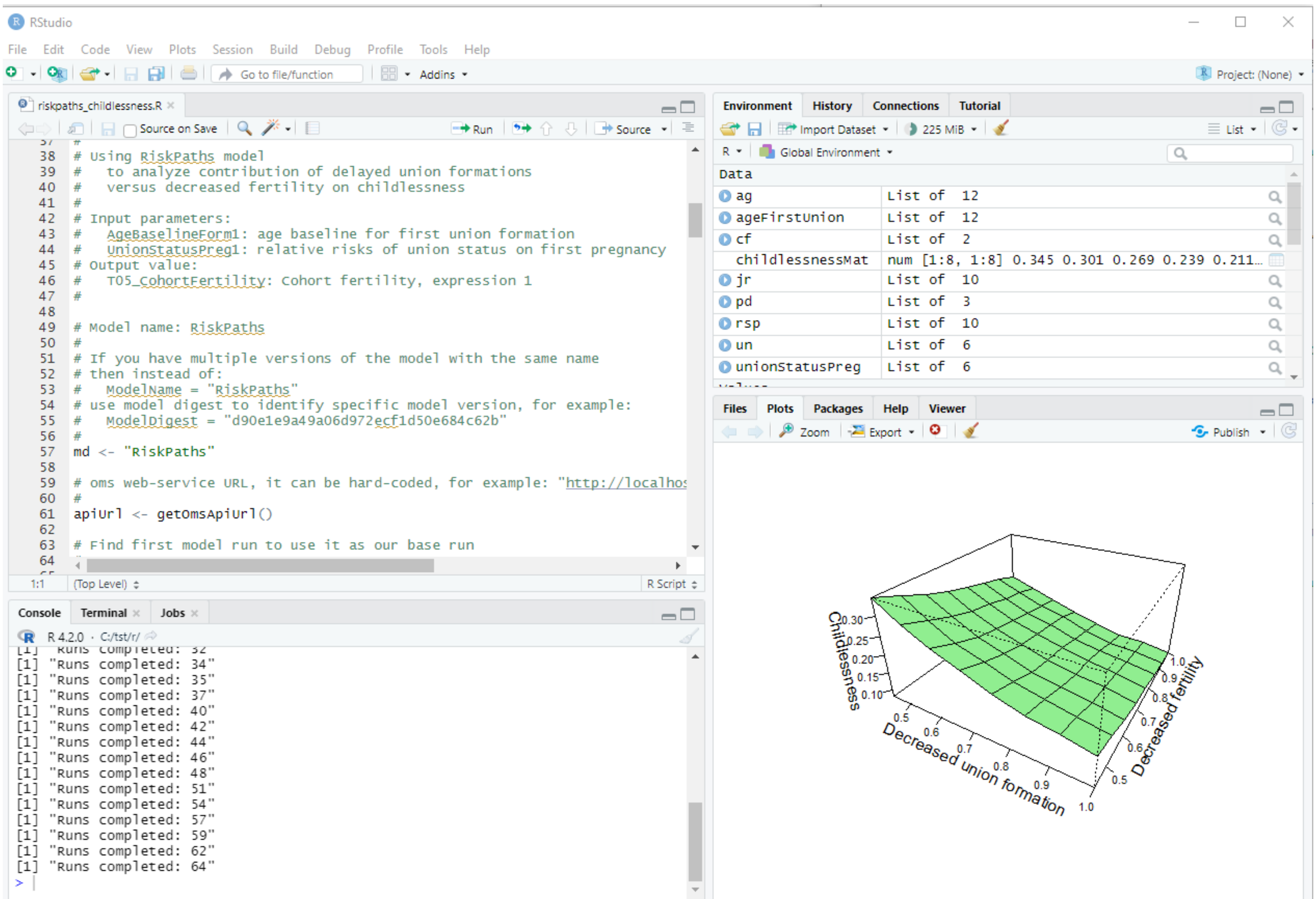
Below is an example of `oms` JSON web-service usage to run RiskPaths model on cloud grid. There is an identical example how to run RiskPaths model on desktop using openMpp package.

Following R example is running "RiskPaths" model to analyze childlessness by varying two parameters:

- Age baseline for first union formation
- Relative risks of union status on first pregnancy by following scale factor:

  ```
  scaleValues <- seq(from = 0.44, to = 1.00, by = 0.08)
  ```

  Please keep in mind, scaling above result in 64 runs of RiskPaths model, to reduce waiting time we are using only 1024 simulation cases in script below.



R script

```
#
# R integration example using RiskPaths model
#   to analyze contribution of delayed union formations
#   versus decreased fertility on childlessness
#
# Prerequisite:
#
# download openM++ release from https://github.com/openmpp/main/releases/latest
# unpack it into any directory
# start oms web-service:
#   Windows:
```

```r
#     cd C:\my-openmpp-release
#     bin\ompp_ui.bat
#   Linux:
#     cd ~/my-openmpp-release
#     bin/oms
#
# Script below is using openM++ web-service "oms"
# to run the model, modify parameters and read output values.
#

# If jsonlite or httr is not installed then do:
#   install.packages("jsonlite")
#   install.packages("httr")
#
library("jsonlite")
library("httr")

# Include openM++ helper functions from your $HOME directory
# on Windows HOME directory is: "C:\Users\User Name Here\Documents"
#
# if you don't have omsCommon.R then download it from https://github.com/openmpp/R/oms-R
# if you have omsCommon.R in some other location then update path below
#
source("~/omsCommon.R")

#
# Using RiskPaths model
#   to analyze contribution of delayed union formations
#   versus decreased fertility on childlessness
#
# Input parameters:
#   AgeBaselineForm1: age baseline for first union formation
#   UnionStatusPreg1: relative risks of union status on first pregnancy
# Output value:
#   T05_CohortFertility: Cohort fertility, expression 1
#

# Model name: RiskPaths
#
# If you have multiple versions of the model with the same name
# then instead of:
#   ModelName = "RiskPaths"
# use model digest to identify specific model version, for example:
#   ModelDigest = "d90e1e9a49a06d972ecf1d50e684c62b"
#
md <- "RiskPaths"

# oms web-service URL, it can be hard-coded, for example: "http://localhost:4040/api/"
#
apiUrl <- getOmsApiUrl()

# Find first model run to use it as our base run
#
# Parameters AgeBaselineForm1 and UnionStatusPreg1 are varied by this script
# and the rest of parameters we are getting from base model run
#
rsp <- GET(paste0(
    apiUrl, "model/", md, "/run/status/first"
  ))
if (http_type(rsp) != 'application/json') {
  stop("Failed to get first run status")
}
jr <- content(rsp)
firstRunDigest <- jr$RunDigest

# get initial values for AgeBaselineForm1 and UnionStatusPreg1 parameters
# by reading it from first model run results
#
rsp <- GET(paste0(
    apiUrl, "model/", md, "/run/", firstRunDigest, "/parameter/AgeBaselineForm1/value/start/0/count/0"
  ))
if (http_type(rsp) != 'application/json') {
  stop("Failed to get parameter AgeBaselineForm1")
}
ageFirstUnion <- content(rsp)

rsp <- GET(paste0(
    apiUrl, "model/", md, "/run/", firstRunDigest, "/parameter/UnionStatusPreg1/value/start/0/count/0"
  ))
if (http_type(rsp) != 'application/json') {
  stop("Failed to get parameter UnionStatusPreg1")
}
unionStatusPreg <- content(rsp)

# Create multiple input scenarios and save all of it as our modelling task:
#   apply scale in range from 0.44 to 1.0
#   to AgeBaselineForm1 and UnionStatusPreg1 parameters
#
```

```
#
# scaleStep <- 0.08 # do 64 model runs
# scaleStep <- 0.5  # use this for quick test
#
scaleStep <- 0.08
scaleValues <- seq(from = 0.44, to = 1.00, by = scaleStep)

nameLst <- c()  # input scenario names, automatically generated

for (scaleAgeBy in scaleValues)
{
  print(c("Scale age: ", scaleAgeBy))

  ag <- ageFirstUnion
  for (k in 1:length(ag))
  {
    ag[[k]]$Value <- ageFirstUnion[[k]]$Value * scaleAgeBy
  }

  for (scaleUnionBy in scaleValues)
  {
    un <- unionStatusPreg
    un[[1]]$Value <- un[[1]]$Value * scaleUnionBy  # change only first two values
    un[[2]]$Value <- un[[2]]$Value * scaleUnionBy  # of UnionStatusPreg1 parameter

    # create new input scenario
    # automatically generate unique names for each input scenario
    #
    pd <- list(
        ModelName = md,
        Name = "",
        BaseRunDigest = firstRunDigest,
        IsReadonly = TRUE,
        Txt = list(
          list(LangCode = "EN", Descr = paste("Scale age:", scaleAgeBy, ", union status", scaleUnionBy)),
          list(LangCode = "FR", Descr = paste("Échelle d'âge:", scaleAgeBy, ", statut syndical", scaleUnionBy))
        ),
        Param = list(
          list(
            Name = "AgeBaselineForm1",
            SubCount = 1,
            Value = ag,
            Txt = list(
              list(LangCode = "FR", Note = paste("Mettre à l'échelle l'âge par:", scaleAgeBy))
            )
          ),
          list(
            Name = "UnionStatusPreg1",
            SubCount = 1,
            Value = un,
            Txt = list(
              list(LangCode = "EN", Note = paste("Scale union status by:", scaleAgeBy))
            )
          )
        )
      )
    jv <- toJSON(pd, pretty = TRUE, auto_unbox = TRUE)

    # create input scenario by submitting request to oms web-service
    rsp <- PUT(paste0(
          apiUrl, "workset-create"
        ),
        body = jv,
        content_type_json()
      )
    if (http_type(rsp) != 'application/json') {
      stop("Failed to create input set")
    }
    jr <- content(rsp)
    sn <- jr$Name  # name of new input scenario generated by oms web-service

    if (is.na(sn) || sn == "") stop("Fail to create input set, scales:", scaleAgeBy, scaleUnionBy)

    nameLst <- c(nameLst, sn)
  }
}

# Create modeling task from all input sets
# automatically generate unique name for the task
#
inpLen <- length(nameLst)

print(paste("Create task from", inpLen, "input scenarios"))

pd <- list(
    ModelName = md,
    Name = "",
    Set = nameLst,
```

```r
    Txt = list(
      list(
        LangCode = "EN",
        Descr = paste("Task to run RiskPaths", inpLen, "times"),
        Note = paste("Task scales AgeBaselineForm1 and UnionStatusPreg1 parameters from 0.44 to 1.00 with step", scaleStep)
      )
    )
  )
jv <- toJSON(pd, pretty = TRUE, auto_unbox = TRUE)


# create task by submitting request to oms web-service
rsp <- PUT(paste0(
      apiUrl, "task-new"
    ),
    body = jv,
    content_type_json()
  )
if (http_type(rsp) != 'application/json') {
  stop("Failed to create modeling task")
}
jr <- content(rsp)
taskName <- jr$Name  # name of new task generated by oms web-service

if (is.na(taskName) || taskName == "") stop("Fail to create modeling task")

#
# Run RiskPaths with modeling task and wait until task is completed
#
# It is MPI model on small computational cluster of 4 servers,
# running 5 RiskPaths_mpi instancces: "root" leader process and 4 computational processes
# each computational process using modelling 4 threads
# root process does only database operations and coordinate child workoload.
#
print(paste("Starting modeling task:", taskName))

pd <- list(
  ModelName = md,
  Mpi = list(
    Np = 5,            # MPI cluster: run 5 processes
    IsNotOnRoot = TRUE, # MPI cluster: do not use root process for modelling
    IsNotByJob = FALSE  # MPI cluster: use job control
  ),
  Template = "mpi.RiskPaths.template.txt",  # MPI cluster: model run tempate
  Opts = list(
    OpenM.TaskName = taskName,
    Parameter.SimulationCases = "1024",    # use 1024 simulation cases to get quick results
    OpenM.BaseRunDigest = firstRunDigest,  # base run to get the rest of input parameters
    OpenM.SubValues = "16",              # use 16 sub-values (sub-samples)
    OpenM.Threads = "4",                 # use 4 modeling threads
    OpenM.ProgressPercent = "100"         # reduce amount of progress messages in the log file
  )
)
jv <- toJSON(pd, pretty = TRUE, auto_unbox = TRUE)

# run modeling task
rsp <- POST(paste0(
      apiUrl, "run"
    ),
    body = jv,
    content_type_json()
  )
if (http_type(rsp) != 'application/json') {
  stop("Failed to run the model")
}
jr <- content(rsp)
taskRunStamp <- jr$RunStamp  # modeling task run stamp

# wait until task completed
runDigests <- waitForTaskCompleted(taskName, taskRunStamp, apiUrl, md)

#
# get results of task run, cohort fertility: T05_CohortFertility.Expr1
#
pd <- list(
    Name = "T05_CohortFertility",
    ValueName = "Expr1",
    Size = 0            # read all rows of T05_CohortFertility.Expr1
  )
jv <- toJSON(pd, pretty = TRUE, auto_unbox = TRUE)

scaleLen <- length(scaleValues)
childlessnessMat <- matrix(data = NA, nrow = scaleLen, ncol = scaleLen, byrow = TRUE)

runIdx <- 1
for (k in 1:scaleLen)
{
  for (i in 1:scaleLen)
```

```
  for (j in 1:scaleLen)
  {
    # for each run digest get T05_CohortFertility.Expr1 value
    #
    rsp <- POST(paste0(
            apiUrl, "model/", md, "/run/", runDigests[runIdx], "/table/value"
        ),
        body = jv,
        content_type_json()
      )
    if (http_type(rsp) != 'application/json') {
      stop("Failed to get T05_CohortFertility.Expr1")
    }
    jt <- content(rsp, type = "text", encoding = "UTF-8")
    cf <- fromJSON(jt, flatten = TRUE)

    # value is not NULL then use it else keep default NA
    if (!cf$Page$IsNull)
    {
      childlessnessMat[k, j] = cf$Page$Value
    }
    runIdx <- runIdx + 1
  }
}

#
# display the results
#
persp(
  x = scaleValues,
  y = scaleValues,
  z = childlessnessMat,
  zlim = range(childlessnessMat, na.rm = TRUE),
  xlab = "Decreased union formation",
  ylab = "Decreased fertility",
  zlab = "Childlessness",
  theta = 30, phi = 30, expand = 0.5, ticktype = "detailed",
  col = "lightgreen",
  cex.axis = 0.7
)
```

Run model from R and save results in CSV file

OpenM++ integration with R: run RiskPaths model on cloud grid

It is a convenient to use GNU R to prepare model parameters and analyze output values. There are two different R APIs which we can use for openM++ models:

- openMpp package: simple and convenient specially for desktop users, upstream and downstream analysis;
- `oms` JSON web-service API: preferable choice to run models on computational clusters and in cloud.

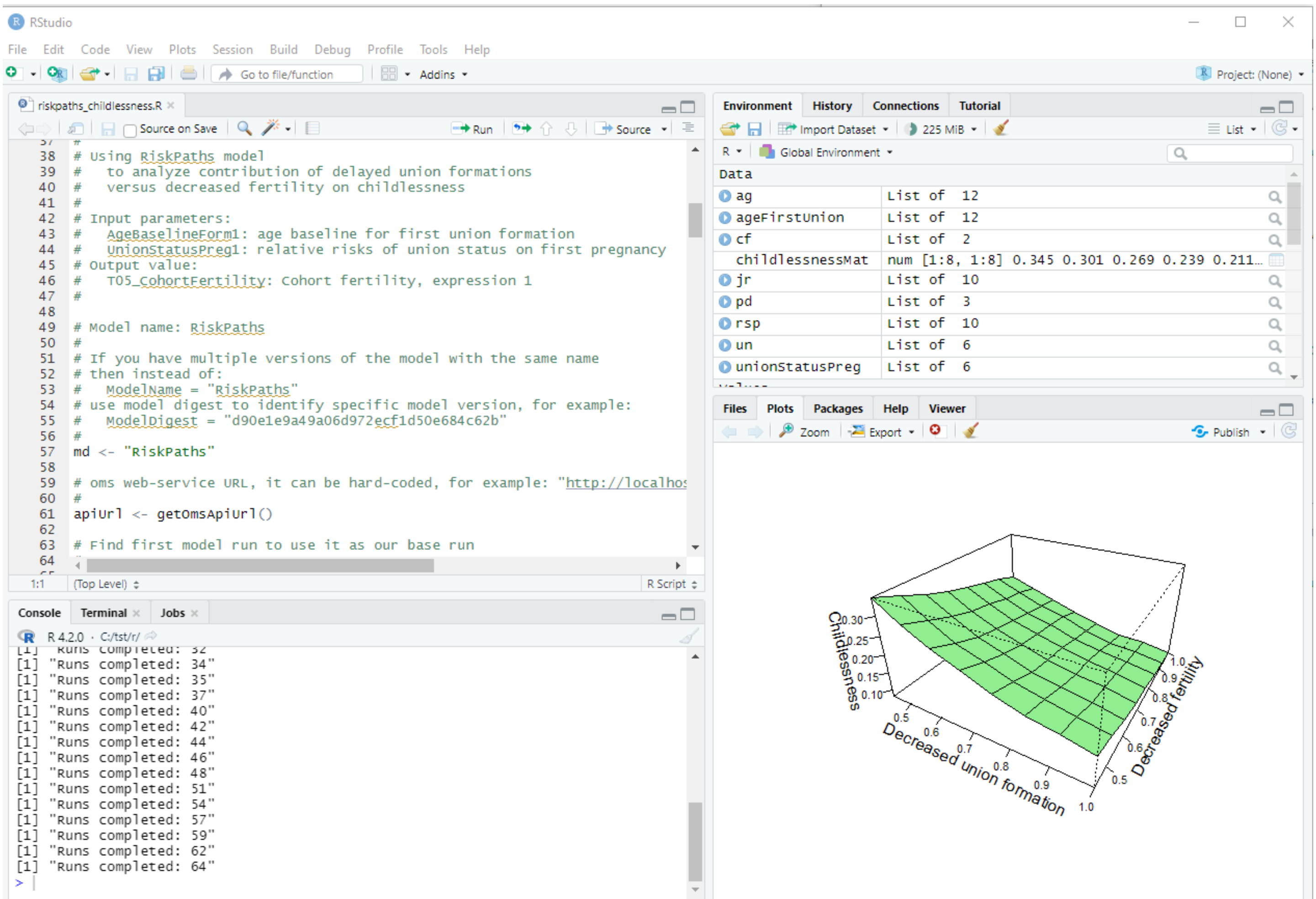
Below is an example of `oms` JSON web-service usage to run RiskPaths model on cloud grid. There is an identical example how to run RiskPaths model on desktop using openMpp package.

Following R example is running "RiskPaths" model to analyze childlessness by varying two parameters:

- Age baseline for first union formation
- Relative risks of union status on first pregnancy by following scale factor:

  ```
  scaleValues <- seq(from = 0.44, to = 1.00, by = 0.08)
  ```

  Please keep in mind, scaling above result in 64 runs of RiskPaths model, to reduce waiting time we are using only 1024 simulation cases in script below.



R script

```
#
# R integration example using RiskPaths model
#   to analyze contribution of delayed union formations
#   versus decreased fertility on childlessness
#
# Prerequisite:
#
# download openM++ release from https://github.com/openmpp/main/releases/latest
# unpack it into any directory
# start oms web-service:
#   Windows:
```

```
#     cd C:\my-openmpp-release
#     bin\ompp_ui.bat
#   Linux:
#     cd ~/my-openmpp-release
#     bin/oms
#
# Script below is using openM++ web-service "oms"
# to run the model, modify parameters and read output values.
#

# If jsonlite or httr is not installed then do:
#   install.packages("jsonlite")
#   install.packages("httr")
#
library("jsonlite")
library("httr")

# Include openM++ helper functions from your $HOME directory
# on Windows HOME directory is: "C:\Users\User Name Here\Documents"
#
# if you don't have omsCommon.R then download it from https://github.com/openmpp/R/oms-R
# if you have omsCommon.R in some other location then update path below
#
source("~/omsCommon.R")

#
# Using RiskPaths model
#   to analyze contribution of delayed union formations
#   versus decreased fertility on childlessness
#
# Input parameters:
#   AgeBaselineForm1: age baseline for first union formation
#   UnionStatusPreg1: relative risks of union status on first pregnancy
# Output value:
#   T05_CohortFertility: Cohort fertility, expression 1
#

# Model name: RiskPaths
#
# If you have multiple versions of the model with the same name
# then instead of:
#   ModelName = "RiskPaths"
# use model digest to identify specific model version, for example:
#   ModelDigest = "d90e1e9a49a06d972ecf1d50e684c62b"
#
md <- "RiskPaths"

# oms web-service URL, it can be hard-coded, for example: "http://localhost:4040/api/"
#
apiUrl <- getOmsApiUrl()

# Find first model run to use it as our base run
#
# Parameters AgeBaselineForm1 and UnionStatusPreg1 are varied by this script
# and the rest of parameters we are getting from base model run
#
rsp <- GET(paste0(
    apiUrl, "model/", md, "/run/status/first"
  ))
if (http_type(rsp) != 'application/json') {
  stop("Failed to get first run status")
}
jr <- content(rsp)
firstRunDigest <- jr$RunDigest

# get initial values for AgeBaselineForm1 and UnionStatusPreg1 parameters
# by reading it from first model run results
#
rsp <- GET(paste0(
    apiUrl, "model/", md, "/run/", firstRunDigest, "/parameter/AgeBaselineForm1/value/start/0/count/0"
  ))
if (http_type(rsp) != 'application/json') {
  stop("Failed to get parameter AgeBaselineForm1")
}
ageFirstUnion <- content(rsp)

rsp <- GET(paste0(
    apiUrl, "model/", md, "/run/", firstRunDigest, "/parameter/UnionStatusPreg1/value/start/0/count/0"
  ))
if (http_type(rsp) != 'application/json') {
  stop("Failed to get parameter UnionStatusPreg1")
}
unionStatusPreg <- content(rsp)

# Create multiple input scenarios and save all of it as our modelling task:
#   apply scale in range from 0.44 to 1.0
#   to AgeBaselineForm1 and UnionStatusPreg1 parameters
#
```

```
#
# scaleStep <- 0.08 # do 64 model runs
# scaleStep <- 0.5  # use this for quick test
#
scaleStep <- 0.08
scaleValues <- seq(from = 0.44, to = 1.00, by = scaleStep)

nameLst <- c()  # input scenario names, automatically generated

for (scaleAgeBy in scaleValues)
{
  print(c("Scale age: ", scaleAgeBy))

  ag <- ageFirstUnion
  for (k in 1:length(ag))
  {
    ag[[k]]$Value <- ageFirstUnion[[k]]$Value * scaleAgeBy
  }

  for (scaleUnionBy in scaleValues)
  {
    un <- unionStatusPreg
    un[[1]]$Value <- un[[1]]$Value * scaleUnionBy  # change only first two values
    un[[2]]$Value <- un[[2]]$Value * scaleUnionBy  # of UnionStatusPreg1 parameter

    # create new input scenario
    # automatically generate unique names for each input scenario
    #
    pd <- list(
        ModelName = md,
        Name = "",
        BaseRunDigest = firstRunDigest,
        IsReadonly = TRUE,
        Txt = list(
          list(LangCode = "EN", Descr = paste("Scale age:", scaleAgeBy, ", union status", scaleUnionBy)),
          list(LangCode = "FR", Descr = paste("Échelle d'âge:", scaleAgeBy, ", statut syndical", scaleUnionBy))
        ),
        Param = list(
          list(
            Name = "AgeBaselineForm1",
            SubCount = 1,
            Value = ag,
            Txt = list(
              list(LangCode = "FR", Note = paste("Mettre à l'échelle l'âge par:", scaleAgeBy))
            )
          ),
          list(
            Name = "UnionStatusPreg1",
            SubCount = 1,
            Value = un,
            Txt = list(
              list(LangCode = "EN", Note = paste("Scale union status by:", scaleAgeBy))
            )
          )
        )
      )
    jv <- toJSON(pd, pretty = TRUE, auto_unbox = TRUE)

    # create input scenario by submitting request to oms web-service
    rsp <- PUT(paste0(
          apiUrl, "workset-create"
        ),
        body = jv,
        content_type_json()
      )
    if (http_type(rsp) != 'application/json') {
      stop("Failed to create input set")
    }
    jr <- content(rsp)
    sn <- jr$Name  # name of new input scenario generated by oms web-service

    if (is.na(sn) || sn == "") stop("Fail to create input set, scales:", scaleAgeBy, scaleUnionBy)

    nameLst <- c(nameLst, sn)
  }
}

# Create modeling task from all input sets
# automatically generate unique name for the task
#
inpLen <- length(nameLst)

print(paste("Create task from", inpLen, "input scenarios"))

pd <- list(
    ModelName = md,
    Name = "",
    Set = nameLst,
```

```
    Txt = list(
      list(
        LangCode = "EN",
        Descr = paste("Task to run RiskPaths", inpLen, "times"),
        Note = paste("Task scales AgeBaselineForm1 and UnionStatusPreg1 parameters from 0.44 to 1.00 with step", scaleStep)
      )
    )
  )
jv <- toJSON(pd, pretty = TRUE, auto_unbox = TRUE)


# create task by submitting request to oms web-service
rsp <- PUT(paste0(
      apiUrl, "task-new"
    ),
    body = jv,
    content_type_json()
  )
if (http_type(rsp) != 'application/json') {
  stop("Failed to create modeling task")
}
jr <- content(rsp)
taskName <- jr$Name  # name of new task generated by oms web-service

if (is.na(taskName) || taskName == "") stop("Fail to create modeling task")

#
# Run RiskPaths with modeling task and wait until task is completed
#
# It is MPI model on small computational cluster of 4 servers,
# running 5 RiskPaths_mpi instancces: "root" leader process and 4 computational processes
# each computational process using modelling 4 threads
# root process does only database operations and coordinate child workoload.
#
print(paste("Starting modeling task:", taskName))

pd <- list(
  ModelName = md,
  Mpi = list(
    Np = 5,             # MPI cluster: run 5 processes
    IsNotOnRoot = TRUE, # MPI cluster: do not use root process for modelling
    IsNotByJob = FALSE  # MPI cluster: use job control
  ),
  Template = "mpi.RiskPaths.template.txt",  # MPI cluster: model run tempate
  Opts = list(
    OpenM.TaskName = taskName,
    Parameter.SimulationCases = "1024",    # use 1024 simulation cases to get quick results
    OpenM.BaseRunDigest = firstRunDigest,  # base run to get the rest of input parameters
    OpenM.SubValues = "16",               # use 16 sub-values (sub-samples)
    OpenM.Threads = "4",                  # use 4 modeling threads
    OpenM.ProgressPercent = "100"         # reduce amount of progress messages in the log file
  )
)
jv <- toJSON(pd, pretty = TRUE, auto_unbox = TRUE)

# run modeling task
rsp <- POST(paste0(
      apiUrl, "run"
    ),
    body = jv,
    content_type_json()
  )
if (http_type(rsp) != 'application/json') {
  stop("Failed to run the model")
}
jr <- content(rsp)
taskRunStamp <- jr$RunStamp  # modeling task run stamp

# wait until task completed
runDigests <- waitForTaskCompleted(taskName, taskRunStamp, apiUrl, md)

#
# get results of task run, cohort fertility: T05_CohortFertility.Expr1
#
pd <- list(
    Name = "T05_CohortFertility",
    ValueName = "Expr1",
    Size = 0            # read all rows of T05_CohortFertility.Expr1
  )
jv <- toJSON(pd, pretty = TRUE, auto_unbox = TRUE)

scaleLen <- length(scaleValues)
childlessnessMat <- matrix(data = NA, nrow = scaleLen, ncol = scaleLen, byrow = TRUE)

runIdx <- 1
for (k in 1:scaleLen)
{
  for (i in 1:scaleLen)
```

```r
  for (j in 1:scaleLen)
  {
    # for each run digest get T05_CohortFertility.Expr1 value
    #
    rsp <- POST(paste0(
            apiUrl, "model/", md, "/run/", runDigests[runIdx], "/table/value"
        ),
        body = jv,
        content_type_json()
      )
    if (http_type(rsp) != 'application/json') {
      stop("Failed to get T05_CohortFertility.Expr1")
    }
    jt <- content(rsp, type = "text", encoding = "UTF-8")
    cf <- fromJSON(jt, flatten = TRUE)

    # value is not NULL then use it else keep default NA
    if (!cf$Page$IsNull)
    {
      childlessnessMat[k, j] = cf$Page$Value
    }
    runIdx <- runIdx + 1
  }
}

#
# display the results
#
persp(
  x = scaleValues,
  y = scaleValues,
  z = childlessnessMat,
  zlim = range(childlessnessMat, na.rm = TRUE),
  xlab = "Decreased union formation",
  ylab = "Decreased fertility",
  zlab = "Childlessness",
  theta = 30, phi = 30, expand = 0.5, ticktype = "detailed",
  col = "lightgreen",
  cex.axis = 0.7
)
```

Windows: Use Docker to get latest version of OpenM++

Why Docker?

There are multiple cases when you want to use Docker containers for openM++ development:

- build your models with latest version of openM++
- build cluster-ready (and cloud-ready) version of your model without installing MPI on your host computer
- do test run of your model in cluster environment without installing and configuring MPI cluster on multiple machines
- build latest version of openM++ from source code without installing and configuring all necessary development tools

All above build and run tasks can be done without Docker and our wiki describes all steps necessary to achieve this. However in that case you will spend a few hours or even days with installing and configuring development and cluster environment. Use of Docker allow to skip unnecessary steps and focus on model development. Also because containers are isolated from host environment there is nothing (except of Docker itself) get installed on your host system and you keep it clean, no software versions conflicts.

In order to use containers Docker for Windows must be installed. It can be done on your host system or on virtual machine. There are short notes about Docker installation at the bottom of that page.

Where to find openM++ Docker images

You can download openM++ images from Docker Hub:

- to run openM++ models pull: `docker pull openmpp/openmpp-run:windows-20H2`
  - Docker Hub description: openmpp/openmpp-run:windows-20H2
  - GitHub: source code and Dockerfile
- to build latest version of openM++ and re-build your models: `docker pull openmpp/openmpp-build:windows-20H2`
  - Docker Hub description: openmpp/openmpp-build:windows-20H2
  - GitHub: source code and Dockerfile

How to use `openmpp/openmpp-run:windows-20H2` to run your models

To run openM++ model do:

```
docker run .... openmpp/openmpp-run:windows-20H2 MyModel.exe
```

For example, if your models are in `C:\my\models\bin` directory then:

```
docker run -v C:\my\models\bin:C:\ompp openmpp/openmpp-run:windows-20H2 MyModel.exe
docker run -v C:\my\models\bin:C:\ompp openmpp/openmpp-run:windows-20H2 mpiexec -n 2 MyModel_mpi.exe -OpenM.SubValues 16
docker run -v C:\my\models\bin:C:\ompp -e OM_ROOT=C:\ompp openmpp/openmpp-run:windows-20H2 MyModel.exe
```

also you can use `-e OM_ROOT=C:\ompp` to set environment variable for your model, if necessary.

To start command prompt do:

```
docker run -v C:\my\models\bin:C:\ompp -it openmpp/openmpp-run:windows-20H2
```

How to use `openmpp/openmpp-build:windows-20H2` to build openM++ and models

To build latest version of openM++ from source code and rebuild your models do:

```
docker run .... openmpp/openmpp-build:windows-20H2 build-all
```

For example, if your build in `C:\my\build` directory then:

```
docker run -v C:\my\build:C:\build openmpp/openmpp-build:windows-20H2 build-all
docker run -v C:\my\build:C:\build -e OM_BUILD_PLATFORMS=x64 openmpp/openmpp-build:windows-20H2 build-all
docker run -v C:\my\build:C:\build -e MODEL_DIRS=RiskPaths   openmpp/openmpp-build:windows-20H2 build-all
```

Following environment variables used to control openM++ build:

```
set OM_BUILD_CONFIGS=Release,Debug (default: Release)
set OM_BUILD_PLATFORMS=Win32,x64   (default: Win32)
set OM_MSG_USE=MPI                 (default: EMPTY)
set MODEL_DIRS=modelOne,NewCaseBased,NewTimeBased,NewCaseBased_bilingual,NewTimeBased_bilingual,IDMM,OzProj,OzProjGen,RiskPaths
```

To build only openM++ libraries and omc compiler do:

```
docker run .... openmpp/openmpp-build:windows-20H2 build-openm
```

Environment variables to control `build-openm: OM_BUILD_CONFIGS, OM_BUILD_PLATFORMS, OM_MSG_USE`

To build models do:

```
docker run .... openmpp/openmpp-build:windows-20H2 build-models
```

Environment variables to control `build-models: OM_BUILD_CONFIGS, OM_BUILD_PLATFORMS, OM_MSG_USE, MODEL_DIRS`

For example, if want to build your own model `MyModel` copy model code into `C:\my\build\models\MyModel` directory and do:

```
docker run -v C:\my\build:C:\build -e MODEL_DIRS=MyModel openmpp/openmpp-build:windows-20H2 build-models
docker run -v C:\my\build:C:\build -e MODEL_DIRS=MyModel -e OM_BUILD_PLATFORMS=x64 openmpp/openmpp-build:windows-20H2 build-models
```

To build openM++ tools do any of:

```
docker run .... openmpp/openmpp-build:windows-20H2 build-go   # Go oms web-service and dbcopy utility
docker run .... openmpp/openmpp-build:windows-20H2 build-r    # openMpp R package
docker run .... openmpp/openmpp-build:windows-20H2 build-perl # Perl utilities
docker run .... openmpp/openmpp-build:windows-20H2 build-ui   # openM++ UI
```

To create `openmpp_win_YYYYMMDD.zip` deployment archive:

```
docker run .... openmpp/openmpp-build:windows-20H2 build-zip
```

Environment variables to control `build-zip: OM_MSG_USE, MODEL_DIRS`

To customize build you can change any of build scripts inside of $HOME/build directory:

```
C:\my\build\build-all.bat     # rebuild entire openM++ and create openmpp_win_YYYYMMDD.tar.gz archive
C:\my\build\build-openm.bat  # rebuild entire openM++ runtime libraries and compiler
C:\my\build\build-models.bat # rebuild openM++ models specified by MODEL_DIRS
C:\my\build\build-go.bat     # rebuild Go oms web-service and dbcopy utility
C:\my\build\build-r.bat      # rebuild openMpp R package
C:\my\build\build-ui.bat     # rebuild openM++ UI
C:\my\build\build-zip.bat    # create openmpp_win_YYYYMMDD.zip archive
```

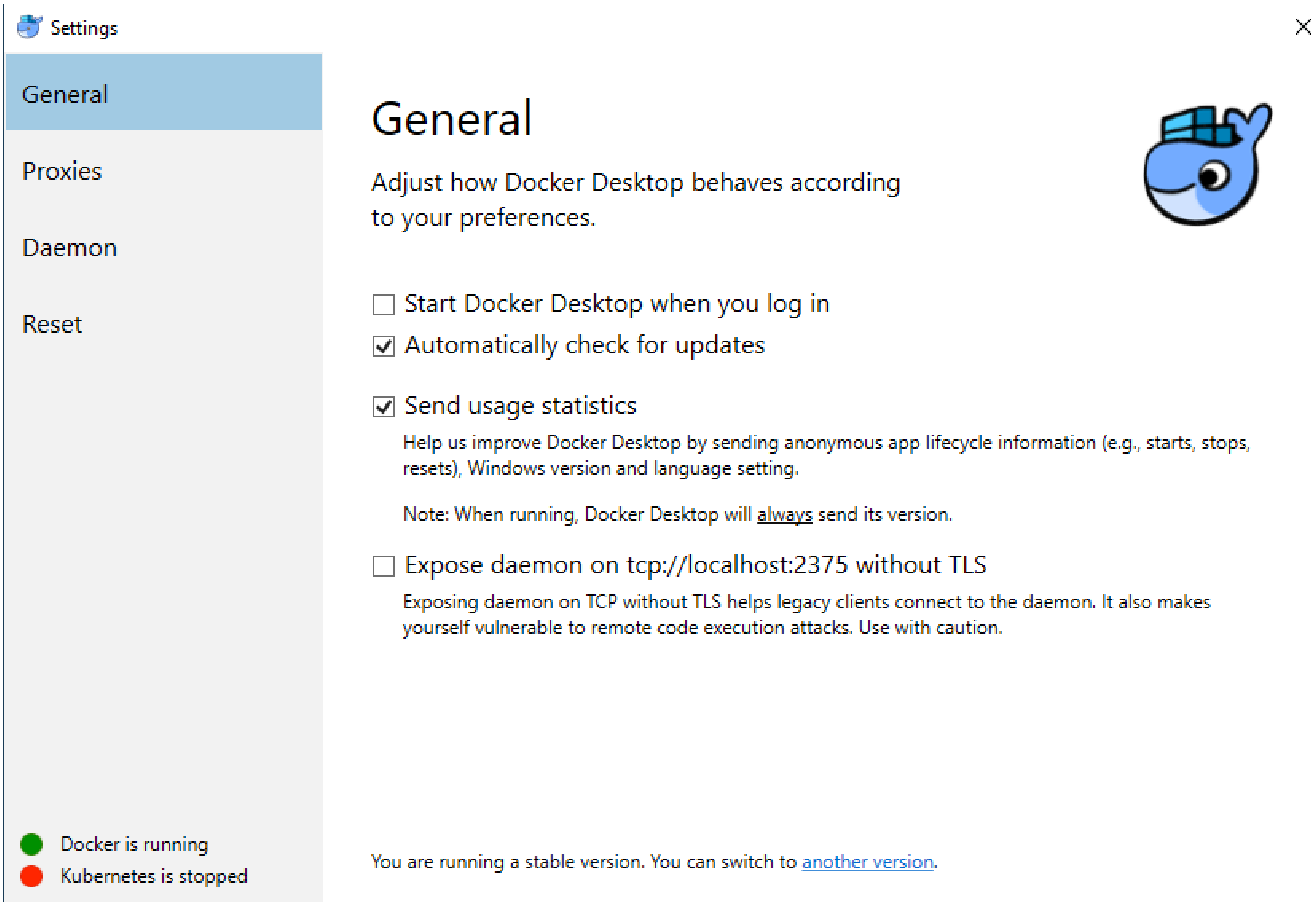
To open cmd command prompt or Perl command prompt:

```
docker run -v C:\my\build:C:\build -it openmpp/openmpp-build:windows-20H2 cmd
docker run -v C:\my\build:C:\build -it openmpp/openmpp-build:windows-20H2 C:\perl\portableshell
```

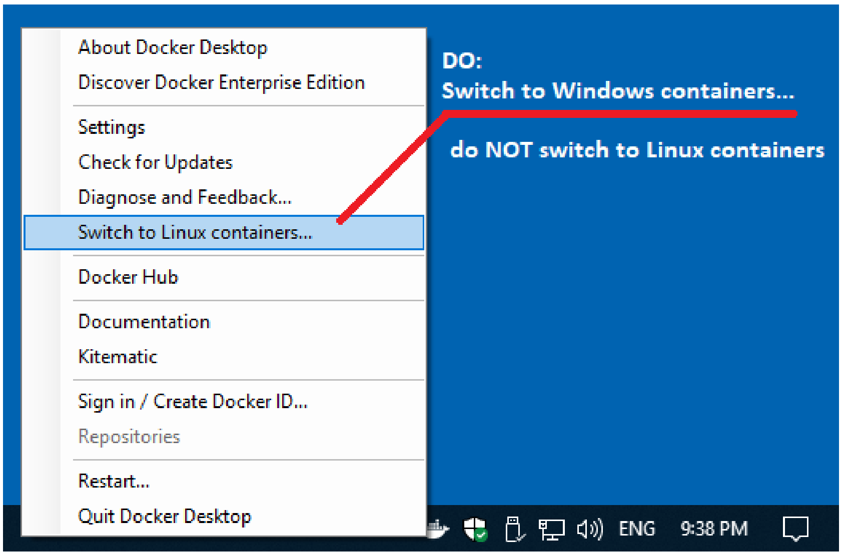
Docker for Windows installation

Please follow official Microsoft documentation and Docker documentation to download and install Docker for Windows. There are few notes below, which you may find useful.

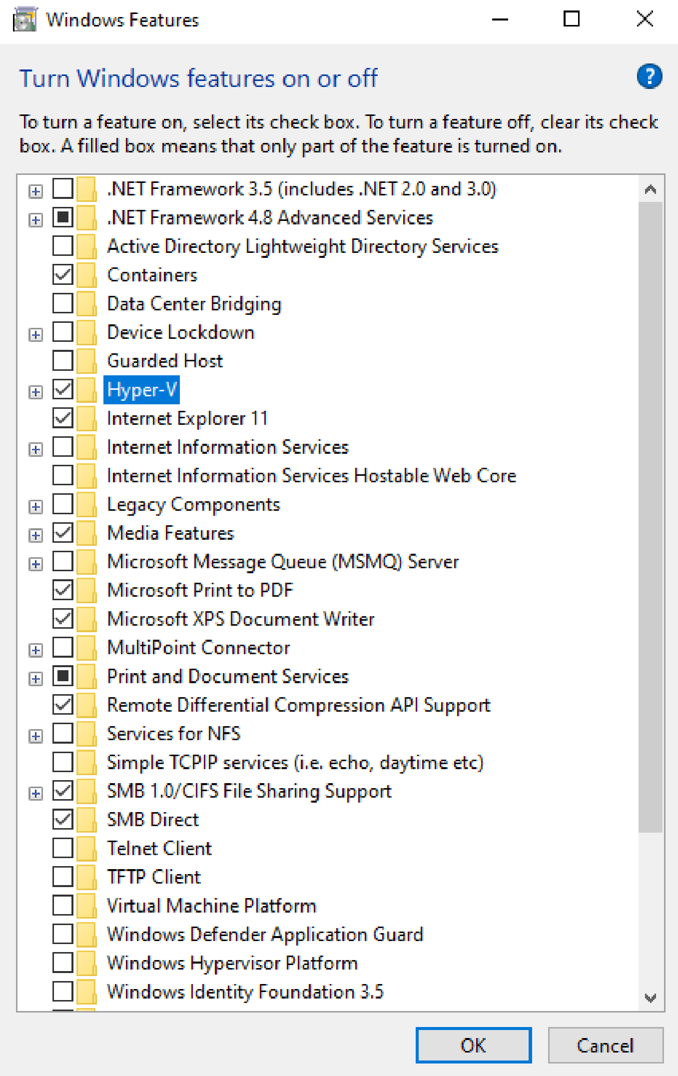
Final result should be "Docker is running":



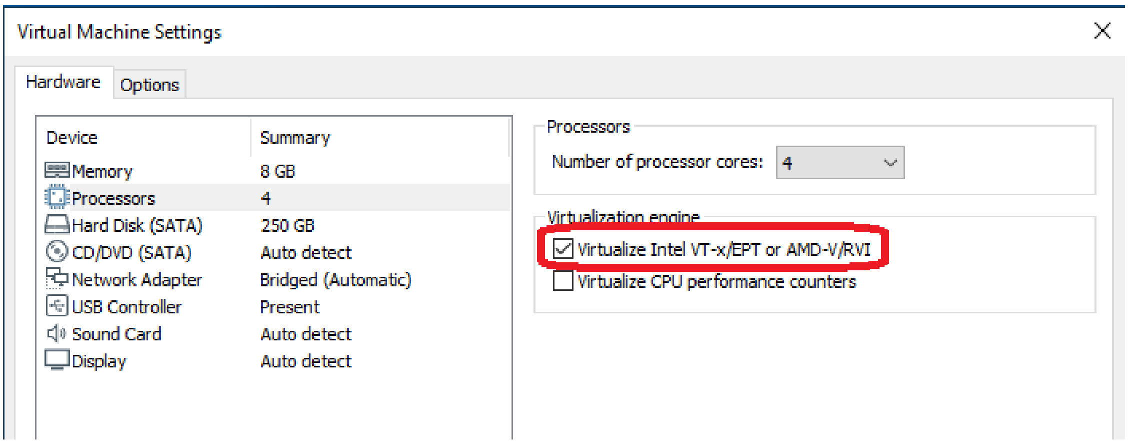
You should do "Switch to Windows containers":



Docker installation require Hyper-V Windows feature "On":



If you installing Docker inside of VMware virtual machine then it may be necessary to turn on "Virtualize Intel VT-x/EPT or AMD-V/RVI" settings. You can turn it off after setup completed:



Linux: Use Docker to get latest version of OpenM++

Why Docker?

There are multiple cases when you want to use Docker containers for openM++ development:

- build your models with latest version of openM++
- build cluster-ready (and cloud-ready) version of your model without installing MPI on your host computer
- do test run of your model in cluster environment without installing and configuring MPI cluster on multiple machines
- build latest version of openM++ from source code without installing and configuring all necessary development tools

All above build and run tasks can be done without Docker and our wiki describes all steps necessary to achieve this. However in that case you will spend a few hours or even days with installing and configuring development and cluster environment. Use of Docker allow to skip unnecessary steps and focus on model development. Also because containers are isolated from host environment there is nothing (except of Docker itself) get installed on your host system and you keep it clean, no software versions conflicts.

To install Docker:

- on Ubuntu do: `sudo apt-get install docker`
- on Debian or MX Linux: `su -c "apt-get install docker"`
- for RedHat 8 please follow RedHat 8: How to use Docker instructions.

Where to find openM++ Docker images

You can download openM++ images from Docker Hub:

- to run openM++ models pull: `docker pull openmpp/openmpp-run:debian`
  - Docker Hub description: openmpp/openmpp-run:debian
  - GitHub: source code and Dockerfile
- to build latest version of openM++ and re-build your models: `docker pull openmpp/openmpp-build:debian`
  - Docker Hub description: openmpp/openmpp-build:debian
  - GitHub: source code and Dockerfile

If you want to use Ubuntu LTS (Ubuntu 20.04):

- to run openM++ models pull: `docker pull openmpp/openmpp-run:ubuntu`
  - Docker Hub description: openmpp/openmpp-run:ubuntu
  - GitHub: source code and Dockerfile
- to build latest version of openM++ and re-build your models: `docker pull openmpp/openmpp-build:ubuntu`
  - Docker Hub description: openmpp/openmpp-build:ubuntu
  - GitHub: source code and Dockerfile

Documentation below contains Debian examples and it is also applicable to Ubuntu. Only difference is `ubuntu` Docker image name and `sudo` to run the `docker` command, for example:

```
sudo docker run openmpp/openmpp-run:ubuntu models/bin/MyModel
```

User name, user ID, group ID, home directory

Both containers `openmpp/openmpp-run:debian` and `openmpp/openmpp-build:debian` created with for user and group `ompp, UID=1999, GID=1999, HOME=/home/ompp`. To avoid permissions issues you may need to do one of of:

- create user `ompp, UID=1999`, group `ompp, UID=1999` and login as that user

- or use `OMPP_*` environment variables as in examples below to map your current login to container

For example, let assume you logged into your system as `user:group = Me:MyGroup UID:GID = 1234:1234` and want to run model in your home directory: `$HOME/models/bin/MyModel`.

Simple attempt to run the model:

```
docker run openmpp/openmpp-run:debian models/bin/MyModel
```

will fail with error similar to: `"models/bin/MyModel: No such file or directory"` because container don't have an access to the files on your host system.

Let's bind your directory `$HOME/models/bin/MyModel` to the container default `/home/ompp`

```
docker run \
  -v $HOME/models/bin:/home/ompp \
  openmpp/openmpp-run:debian \
  ./MyModel
```

That will fail with error `"Permission denied"` because container default login `user:group = ompp:ompp UID:GID = 1999:1999` don't have an access to your files `user:group = Me:MyGroup UID:GID = 1234:1234`.

You can create such login on your host system `user:group = ompp:ompp UID:GID = 1999:1999` and use it to run the models

Or you can tell container to use your current `user:group = Me:MyGroup UID:GID = 1234:1234` instead of default values:

```
docker run \
  -v $HOME/models/bin:/home/models \
  -e OMPP_USER=models -e OMPP_GROUP=models -e OMPP_UID=$UID -e OMPP_GID=`id -g` \
  openmpp/openmpp-run:debian \
  ./MyModel
```

How to use `openmpp/openmpp-run:debian` to run your models

To run openM++ model do:

```
docker run .... openmpp/openmpp-run:debian ./MyModel
```

For example, if your models are in `$HOME/models/bin` directory then:

```
docker run \
  -v $HOME/models/bin:/home/models \
  -e OMPP_USER=models -e OMPP_GROUP=models -e OMPP_UID=$UID -e OMPP_GID=`id -g` \
  openmpp/openmpp-run:debian \
  ./MyModel

docker run \
  -v $HOME/models/bin:/home/models \
  -e OMPP_USER=models -e OMPP_GROUP=models -e OMPP_UID=$UID -e OMPP_GID=`id -g` \
  openmpp/openmpp-run:debian \
  mpiexec -n 2 MyModel_mpi -OpenM.SubValues 16
```

also you can use `-e OM_ROOT=/home/ompp` to set environment variable for your model, if necessary.

To start shell inside of conatiner do:

```
docker run \
  -v $HOME:/home/${USER} \
  -e OMPP_USER=${USER} -e OMPP_GROUP=`id -gn` -e OMPP_UID=$UID -e OMPP_GID=`id -g` \
  -it openmpp/openmpp-run:debian
  bash
```

Following environment variables are used to map container user to your login:

```
OMPP_USER=ompp   # default: ompp, container user name and HOME
OMPP_GROUP=ompp  # default: ompp, container group name
OMPP_UID=1999    # default: 1999, container user ID
OMPP_GID=1999    # default: 1999, container group ID
```

How to use `openmpp/openmpp-build:debian` to build openM++ and models

To build latest version of openM++ from source code and rebuild your models do:

```
docker run ....options.... openmpp/openmpp-build:debian ./build-all
```

For example, if your build in `$HOME/build` directory then:

```
docker run \
  -v $HOME/build:/home/build \
  -e OMPP_USER=build -e OMPP_GROUP=build -e OMPP_UID=$UID -e OMPP_GID=`id -g` \
  openmpp/openmpp-build:debian \
  ./build-all

docker run \
  -v $HOME/build_mpi:/home/build_mpi \
  -e OMPP_USER=build_mpi -e OMPP_GROUP=build_mpi -e OMPP_UID=$UID -e OMPP_GID=`id -g` \
  -e OM_MSG_USE=MPI \
  openmpp/openmpp-build:debian \
  ./build-all

docker run ....user, group, home.... -e MODEL_DIRS=RiskPaths,IDMM      openmpp/openmpp-build:debian ./build-all
docker run ....user, group, home.... -e OM_BUILD_CONFIGS=RELEASE,DEBUG openmpp/openmpp-build:debian ./build-all
docker run ....user, group, home.... -e OM_MSG_USE=MPI                 openmpp/openmpp-build:debian ./build-all
```

Following environment variables used to control openM++ build:

```
OM_BUILD_CONFIGS=RELEASE,DEBUG # default: RELEASE,DEBUG for libraries and RELEASE for models
OM_MSG_USE=MPI                 # default: EMPTY
MODEL_DIRS=modelOne,NewCaseBased,NewTimeBased,NewCaseBased_bilingual,NewTimeBased_bilingual,IDMM,OzProj,OzProjGen,RiskPaths
```

Following environment variables are used to map container user to your login:

```
OMPP_USER=ompp   # default: ompp, container user name and HOME
OMPP_GROUP=ompp  # default: ompp, container group name
OMPP_UID=1999    # default: 1999, container user ID
OMPP_GID=1999    # default: 1999, container group ID
```

To build only openM++ libraries and omc compiler do:

```
docker run ....options.... openmpp/openmpp-build:debian ./build-openm
```

Environment variables to control `build-openm: OM_BUILD_CONFIGS, OM_MSG_USE`

To build only models do:

```
docker run ....options.... openmpp/openmpp-build:debian ./build-models
```

Environment variables to control `build-models: OM_BUILD_CONFIGS, OM_MSG_USE, MODEL_DIRS`

For example, if want to build your own model `MyModel` copy model code into `$HOME/build/models/MyModel` directory and do:

```
docker run ....user, group, home.... -e MODEL_DIRS=MyModel openmpp/openmpp-build:debian ./build-models
docker run ....user, group, home.... -e MODEL_DIRS=MyModel -e OM_BUILD_CONFIGS=RELEASE,DEBUG openmpp/openmpp-build:debian ./build-models
```

To build openM++ tools do any of:

```
docker run .... openmpp/openmpp-build:debian ./build-go  # Go oms web-service and dbcopy utility
docker run .... openmpp/openmpp-build:debian ./build-r   # openMpp R package
docker run .... openmpp/openmpp-build:debian ./build-ui  # openM++ UI
```

To create `openmpp_redhat_YYYYMMDD.tar.gz` deployment archive:

```
docker run .... openmpp/openmpp-build:debian ./build-tar-gz
```

Environment variables to control `build-tar-gz: OM_MSG_USE, MODEL_DIRS`

To customize build you can change any of build scripts inside of `$HOME/build` directory:

```
$HOME/build/build-all     # rebuild entire openM++ and create openmpp_redhat_YYYYMMDD.tar.gz archive
$HOME/build/build-openm   # rebuild entire openM++ runtime libraries and compiler
$HOME/build/build-models  # rebuild openM++ models specified by MODEL_DIRS
$HOME/build/build-go      # rebuild Go oms web-service and dbcopy utility
$HOME/build/build-r       # rebuild openMpp R package
$HOME/build/build-ui      # rebuild openM++ UI
$HOME/build/build-tar-gz  # create openmpp_redhat_YYYYMMDD.tar.gz archive
```

To start shell inside of container do:

```
docker run \
  -v $HOME:/home/${USER} \
  -e OMPP_USER=${USER} -e OMPP_GROUP=`id -gn` -e OMPP_UID=$UID -e OMPP_GID=`id -g` \
  -it openmpp/openmpp-build:debian \
  bash
```

How to use `openmpp/openmpp-build:debian` to update openM++ documentation

To build latest version of openM++ documentation do:

```
docker run ....options.... openmpp/openmpp-build:debian ./make-doc
```

For example, if your want to make a documenation in `$HOME/build_doc` directory then:

```
docker run \
  -v $HOME/build_doc:/home/build_doc \
  -e OMPP_USER=build_doc -e OMPP_GROUP=build_doc -e OMPP_UID=$UID -e OMPP_GID=`id -g` \
  openmpp/openmpp-build:debian \
  ./make-doc
```

RedHat 8: Use Docker to get latest version of OpenM++

Why Docker?

There are multiple cases when you want to use Docker containers for openM++ development:

- build your models with latest version of openM++
- build cluster-ready (and cloud-ready) version of your model without installing MPI on your host computer
- do test run of your model in cluster environment without installing and configuring MPI cluster on multiple machines
- build latest version of openM++ from source code without installing and configuring all necessary development tools

All above build and run tasks can be done without Docker and our wiki describes all steps necessary to achieve this. However in that case you will spend a few hours or even days with installing and configuring development and cluster environment. Use of Docker allow to skip unnecessary steps and focus on model development. Also because containers are isolated from host environment there is nothing (except of Docker itself) get installed on your host system and you keep it clean, no software versions conflicts.

To install Docker on RedHat do: `dnf install podman`

Where to find openM++ Docker images

You can download openM++ images from Docker Hub:

- to run openM++ models pull: `podman pull openmpp/openmpp-run:redhat-8`
  - Docker Hub description: openmpp/openmpp-run:redhat-8
  - GitHub: source code and Dockerfile
- to build latest version of openM++ and re-build your models: `podman pull openmpp/openmpp-build:redhat-8`
  - Docker Hub description: openmpp/openmpp-build:redhat-8
  - GitHub: source code and Dockerfile

User name and home directory

Both containers `openmpp/openmpp-run:redhat-8` and `openmpp/openmpp-build:redhat-8` created with for user `ompp, HOME=/home/ompp`. To avoid permissions issues you may need to map that user to your host user namespace and use `:z` option if you want to mount host local directory, for example:

```
podman run -userns=host -v $HOME/build:/home/build:z -e OMPP_USER=build openmpp/openmpp-build:redhat-8 ./build-all
```

Above we are mapping container user `build` to our current host user and container user home directory `/home/build` to sub-folder `$HOME/build`.

Or if want to use container user `models` to run our models:

```
podman run -userns=host -v $HOME/models:/home/models:z -e OMPP_USER=models openmpp/openmpp-run:redhat-8 ./modelOne
```

How to use `openmpp/openmpp-run:redhat-8` to run your models

To run openM++ model do:

```
podman run .... openmpp/openmpp-run:redhat-8 ./modelOne
```

For example, if your models are in `$HOME/models/bin` directory then:

```
podman run \
  -userns=host \
  -v $HOME/models/bin:/home/models:z \
  -e OMPP_USER=models \
  openmpp/openmpp-run:redhat-8 \
  ./modelOne

podman run \
  -userns=host \
  -v $HOME/models/bin:/home/models:z \
  -e OMPP_USER=models \
  openmpp/openmpp-run:redhat-8 \
  mpiexec --allow-run-as-root -n 2 MyModel_mpi -OpenM.SubValues 16
```

Also you can use `-e OM_ROOT=/home/models/my-openMpp-dir` to set environment variable for your model, if necessary.

To start shell inside of conatiner do:

```
podman run -it openmpp/openmpp-run:redhat-8 bash
```

Following environment variable is used to map container user and home directory to you host directory:

```
OMPP_USER=ompp   # default: ompp, container user name and HOME
```

How to use `openmpp/openmpp-build:redhat-8` to build openM++ and models

To build latest version of openM++ from source code and rebuild your models do:

```
podman run .... openmpp/openmpp-build:redhat-8 ./build-all
```

For example, if your build in `$HOME/build` or in `$HOME/build_mpi` directory then:

```
podman run \
  -userns=host \
  -v $HOME/build:/home/build:z \
  -e OMPP_USER=build \
  openmpp/openmpp-build:redhat-8 \
  ./build-all

podman run \
  -userns=host \
  -v $HOME/build_mpi:/home/build_mpi:z \
  -e OMPP_USER=build_mpi \
  -e OM_MSG_USE=MPI \
  openmpp/openmpp-build:redhat-8 \
  ./build-all

podman run .... -e MODEL_DIRS=RiskPaths,IDMM       openmpp/openmpp-build:redhat-8 ./build-all
podman run .... -e OM_BUILD_CONFIGS=RELEASE,DEBUG openmpp/openmpp-build:redhat-8 ./build-all
podman run .... -e OM_MSG_USE=MPI                  openmpp/openmpp-build:redhat-8 ./build-all
```

Following environment variables used to control openM++ build:

```
OM_BUILD_CONFIGS=RELEASE,DEBUG # default: RELEASE,DEBUG for libraries and RELEASE for models
OM_MSG_USE=MPI                 # default: EMPTY
MODEL_DIRS=modelOne,NewCaseBased,NewTimeBased,NewCaseBased_bilingual,NewTimeBased_bilingual,IDMM,OzProj,OzProjGen,RiskPaths
```

Following environment variable is used to map container user and home directory to you host directory:

```
OMPP_USER=ompp   # default: ompp, container user name and HOME
```

To build only openM++ libraries and omc compiler do:

```
podman run .... openmpp/openmpp-build:redhat-8 ./build-openm
```

Environment variables to control `build-openm: OM_BUILD_CONFIGS, OM_MSG_USE`

To build only models do:

```
podman run .... openmpp/openmpp-build:redhat-8 ./build-models
```

Environment variables to control `build-models: OM_BUILD_CONFIGS, OM_MSG_USE, MODEL_DIRS`

For example, if want to build your own model `MyModel` copy model code into `$HOME/build/models/MyModel` directory and do:

```
podman run .... openmpp/openmpp-build:redhat-8 ./build-models
podman run .... -e MODEL_DIRS=MyModel openmpp/openmpp-build:redhat-8 ./build-models
podman run .... -e MODEL_DIRS=MyModel -e OM_BUILD_CONFIGS=RELEASE,DEBUG openmpp/openmpp-build:redhat-8 ./build-models
```

To build openM++ tools do any of:

```
podman run .... openmpp/openmpp-build:redhat-8 ./build-go   # Go oms web-service and dbcopy utility
podman run .... openmpp/openmpp-build:redhat-8 ./build-r    # openMpp R package
podman run .... openmpp/openmpp-build:redhat-8 ./build-ui   # openM++ UI
```

To create `openmpp_redhat_YYYYMMDD.tar.gz` deployment archive:

```
podman run .... openmpp/openmpp-build:redhat-8 ./build-tar-gz
```

Environment variables to control `build-tar-gz: OM_MSG_USE, MODEL_DIRS`

To customize build you can change any of build scripts inside of `$HOME/build` directory:

```
$HOME/build/build-all     # rebuild entire openM++ and create openmpp_redhat_YYYYMMDD.tar.gz archive
$HOME/build/build-openm   # rebuild entire openM++ runtime libraries and compiler
$HOME/build/build-models  # rebuild openM++ models specified by MODEL_DIRS
$HOME/build/build-go      # rebuild Go oms web-service and dbcopy utility
$HOME/build/build-r       # rebuild openMpp R package
$HOME/build/build-ui      # rebuild openM++ UI
$HOME/build/build-tar-gz  # create openmpp_redhat_YYYYMMDD.tar.gz archive
```

To start shell inside of container do:

```
podman run -it openmpp/openmpp-build:redhat-8 bash
```

Quick Start for OpenM++ Developers

Where is OpenM++

- Download: binary files and source code
- Latest source code: openM++ git
- (optional) Go source code: openM++ Go git
- (optional) UI source code: openM++ Go git
- Documentation: this wiki
- Pre-requisits described at: Setup Development Environment.

It is recommended to start from desktop version of openM++, not a cluster (MPI) version.

You need to use cluster version of openM++ to run the model on multiple computers in your network, in cloud or HPC cluster environment. OpenM++ is using MPI to run the models on multiple computers. Please check Model Run: How to Run the Model page for more details.

Build on Linux

Tested platforms:

- Debian 10 and 11, MX Linux 19 and 21, Ubuntu 20.04, RedHat 8
- g++ >= 8.3
- (optional) MPI, i.e.: OpenMPI >= 3.1 or MPICH (other MPI implementations expected to work but not tested)
- (optional) OpenMPI >= 4.0 on RedHat >= 8.3 (OpenMPI was broken on RedHat 8.1)

It is not supported, but may also work on older versions, for example Ubuntu 18.04.

Check your `g++ --version`:

```
g++ (Debian 8.3.0-6) 8.3.0                # Debian 10 and MX Linux 19
g++ (Debian 10.2.1-6) 10.2.1 20210110     # Debian 11 and MX Linux 21
g++ (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0  # Ubuntu 20.04
g++ (GCC) 8.3.1 20191121 (Red Hat 8.3.1-5) # RedHat 8, Rocky Linux, AlmaLinux
```

To build **debug version** of openM++:

```
git clone https://github.com/openmpp/main.git master
cd master/openm/
make
cd ../models/
make
```

RedHat 8: If want to rebuild omc (OpenM++ compiler) then you will need `bison` version 3.3+ and `flex` 2.6+ installed, see details at: Setup Development Environment. It is optional and you can avoid it by rebuilding only openM++ run-time libararies:

```
git clone https://github.com/openmpp/main.git master
cd master/openm/
make libopenm
cd ../models/
make
```

To build **release version** of openM++: `make RELEASE=1`

To build **MPI version** of openM++: `make OM_MSG_USE=MPI`

Note: openM++ binary downloads build as: `make RELEASE=1 OM_MSG_USE=MPI`

RedHat 8: to build and run MPI version of openM++:

```
module load mpi/openmpi-x86_64
```

Of course, you can also use 32bit version of OpenMPI or MPICH.

Build on Windows

Tested platforms:

- Windows 10, Windows 7 (64 and 32 bits), 2016 (64 bit)
- expected to work on any Windows 7 and above or 2008R2 and above, 32 and 64 bits, not regulary tested
- Visual Studio 2019 or 2017 (VS 2017 not supported, but may work), including Community Edition
- (optional) Microsoft MPI SDK Redistributable Package

To build **debug version** of openM++:

- checkout from openM++ git using your favorite Git client into `C:\SomeDir\` or use command line:

  ```
  git clone https://github.com/openmpp/main.git SomeDir
  ```

- download and unzip Windows version of bison and flex into `C:\SomeDir\bin\`.
- download and unzip sqlite3.exe into `C:\SomeDir\bin\`.
- use Visual Studio or MSBuild to build `C:\SomeDir\openm\openm.sln` solution.
- to build test model(s), i.e.: NewCaseBased, use Visual Studio or MSBuild: `C:\SomeDir\models\NewCaseBased\NewCaseBased-ompp.sln`.

To build **MPI version** of openM++:

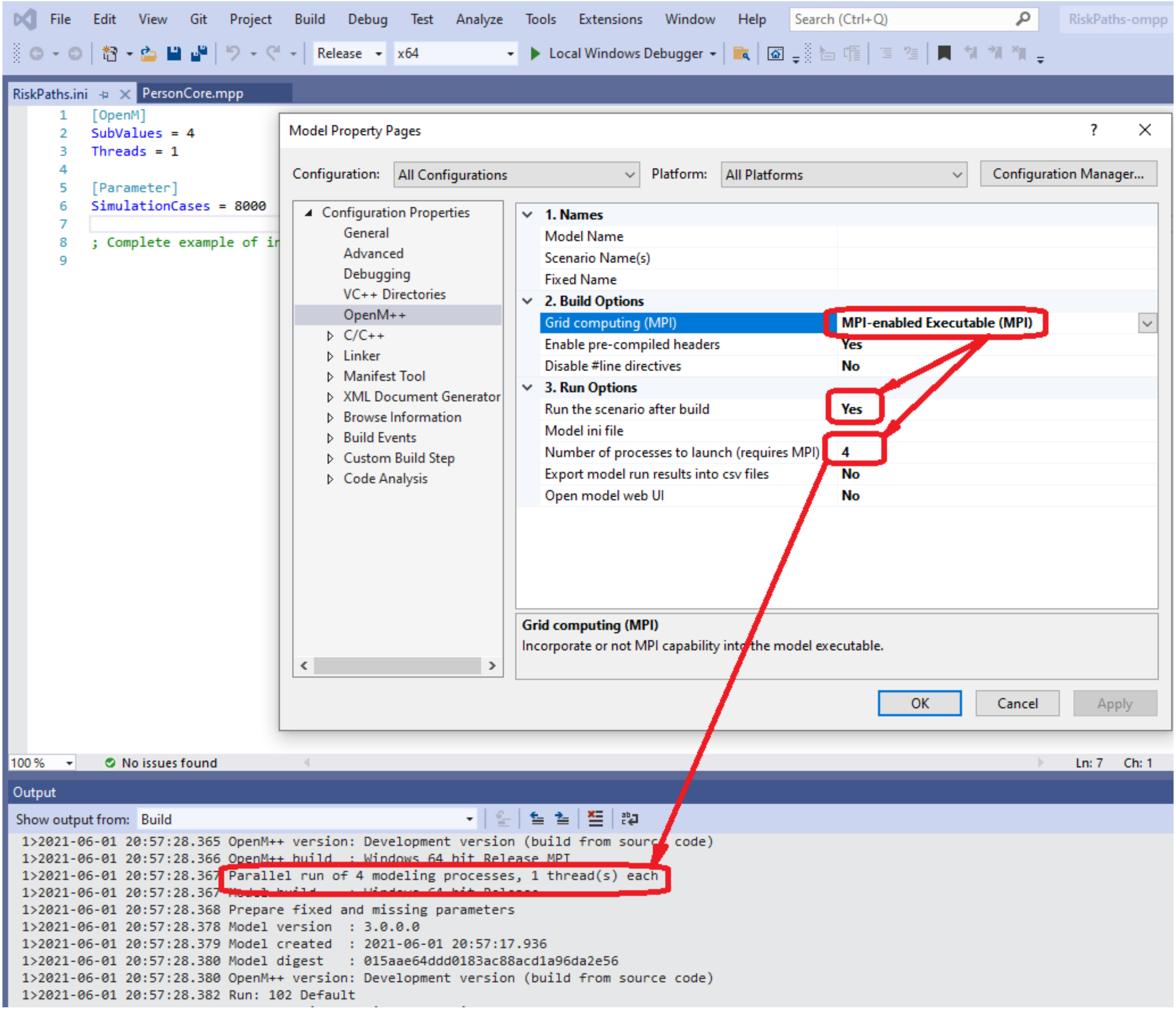
- download and install Microsoft MPI SDK and MPI Redistributable.
- use Notepad to open `C:\SomeDir\openm\openm.build.props`, find and edit the line:

  ```
  <OM_MSG_USE>MPI</OM_MSG_USE>
  ```

- build `C:\SomeDir\openm\openm.sln` solution.
- rebuild the model and run it:
  - go to menu: Project -> Properties -> Configuration Properties -> OpenM++
  - change: Build Options -> Grid computing (MPI) -> MPI-enabled Executable (MPI)
  - change: Run Options -> Number of processes to launch -> ....2 or more (depends on your cluster configuration)...
  - change: Run Options -> Run the scenario after build -> Yes
  - Rebuild Model project

At bottom Output window of Visual Studio you will see something like:

```
1>Model.vcxproj -> C:\SomeDir\models\RiskPaths\ompp\bin\RiskPaths_mpi.exe
1>2021-06-01 20:57:28.146 RiskPaths
1>2021-06-01 20:57:28.146 RiskPaths
1>2021-06-01 20:57:28.146 RiskPaths
1>2021-06-01 20:57:28.163 RiskPaths
........
1>2021-06-01 20:57:28.366 OpenM++ build  : Windows 64 bit Release MPI
1>2021-06-01 20:57:28.367 Parallel run of 4 modeling processes, 1 thread(s) each
........
1>2021-06-01 20:57:28.859 member=3 Simulation progress=100% cases=2000
1>2021-06-01 20:57:28.867 member=3 Simulation summary: cases=2000, events/case=112.9, entities/case=1.0, elapsed=0.453989s
1>2021-06-01 20:57:28.868 member=3 Write output tables - start
1>2021-06-01 20:57:28.873 member=3 Write output tables - finish
1>2021-06-01 20:57:29.233 member=0 Write output tables - finish
1>2021-06-01 20:57:29.919 Writing into aggregated output tables, run: 102
1>2021-06-01 20:57:32.607 Done.
1>2021-06-01 20:57:32.607 Done.
1>2021-06-01 20:57:32.607 Done.
1>2021-06-01 20:57:32.607 Done.
1>Done building project "Model.vcxproj".
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========
```



Note: binary downloads build with Microsoft MPI SDK and MPI Redistributable.

Note: If you getting build error MSB8036:

```
C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\Common7\IDE\VC\VCTargets\Platforms\Win32\PlatformToolsets\v141\Toolset.targets(34,5):
error MSB8036: The Windows SDK version 10.0.14393.0 was not found.
Install the required version of Windows SDK or change the SDK version in the project property pages or by right-clicking the solution and selecting "Retarget solution".
```

then do one of the following:

- "Retarget solution"
- use Visual Studio 2019
- start Visual Studio 2017 Installer (VS 2017 not supported but may work)
  - Modify
  - right column
  - check box Windows 8.1 SDK and UCRT SDK

Build on MacOS

- Tested on MacOS 10.15 Catalina and 11.1+ Big Sur

Check your clang, make, bison, SQLite version:

```
clang --version
....
Apple clang version 11.0.0 (clang-1100.0.33.12)

make --version
....
GNU Make 3.81

bison --version
....
bison (GNU Bison) 3.8.2

sqlite3 --version
....
3.28.0 2019-04-15 14:49:49
```

To build **debug version** of openM++:

```
git clone https://github.com/openmpp/main.git ompp-main
cd ompp-main/openm/
make
cd ../models/
make
```

To build **release version** of openM++: `make RELEASE=1`

You can also use Xcode `~/ompp-main/openm/openm.xcworkspace`.

In order to build omc complier you need to use menu and select Product -> Scheme -> omc

Known issue: Xcode UI does not update check mark on selected scheme To fix it go to Product -> Scheme -> Manage Schemes and use mouse to drag any scheme to move it up or down.

Release version of omc is required in order to build any model other than modelOne.

In order to build and debug modelOne using Xcode please open `~/ompp-main/models/modelOne/modelOne.xcworkspace`

Build R package

- clone from GitHub:

  ```
  git clone https://github.com/openmpp/R.git ompp-r
  ```

- Windows:

  ```
  cd C:>C:\ompp-r
  "C:\Program Files\R\R-3.4.0\bin\R.exe" CMD build openMpp
  ```

- Linux and MacOS:

```
cd ompp-r
R CMD build openMpp
```

Expected output:

```
* checking for file 'openMpp/DESCRIPTION' ... OK
* preparing 'openMpp':
* checking DESCRIPTION meta-information ... OK
* checking for LF line-endings in source and make files and shell scripts
* checking for empty or unneeded directories
* building 'openMpp_0.8.3.tar.gz'
```

Build Go utilities

- setup Go envirnment as described at: Setup Development Environment.
- initial checkout:

  ```
  mkdir $HOME/ompp
  cd $HOME/ompp
  export GOPATH=$HOME/ompp
  git clone https://github.com/openmpp/go ompp-go
  ```

- build Go utilities:

  ```
  cd ompp-go
  go get github.com/openmpp/go/dbcopy
  go get github.com/openmpp/go/oms
  ```

- rebuild Go utilities if you change source code:

  ```
  cd $HOME/ompp/ompp-go
  go install github.com/openmpp/go/dbcopy
  go install github.com/openmpp/go/oms
  ```

After initial checkout first `go get` command can take ~30 seconds because go needs to get all dependencies. To see more deatils you can use:

```
go get -v github.com/openmpp/go/dbcopy
```

By default only SQLite model databases supported by `dbcopy` and `oms`. If you want to use other databases vendors please compile `dbcopy` with ODBC enabled:

```
go install -tags odbc github.com/openmpp/go/dbcopy
```

Currently supported database vendors are: SQLite (default), Microsoft SQL Server, MySql, PostgreSQL, IBM DB2, Oracle. You can use dbcopy utility to copy model data between any of vendors above, for example copy from MySQL to MSSQL or from PostgeSQL to SQLite.

Build UI

Instructions below assuming Windows environment and it is very much identical for Linux and MacOS, except of course, back slashes in directory paths.

- setup `node.js` environment as described at: Setup Development Environment.
- checkout and build UI:

  ```
  cd my-openm-plus-plus-dir
  git clone https://github.com/openmpp/UI.git ompp-ui
  cd ompp-ui
  npm install
  ```

- make sure you have `models\bin` populated with *.sqlite db files and model executables.
- it is recommended to have `my-openm-plus-plus-dir\etc` folder which can be found at openM++ release archive
- start oms web-service by invoking:
  - `ompp_ui.bat` on Windows

- `ompp_ui.sh` on Linux
- `ompp_ui.command` on MacOS
- or do it in command line:

  ```
  cd my-openmn-plus-plus-dir
  bin\oms -oms.HomeDir models -oms.LogRequest
  ```

- start UI in debug mode:

  ```
  cd my-openm-plus-plus-dir\ompp-ui
  npm run dev
  ```

- open your favorite browser at `http://localhost:8080`
- to build UI for production:

  ```
  cd my-openm-plus-plus-dir\ompp-ui
  npm run build
  ```

- copy HTML results folder `my-openm-plus-plus-dir\dist\spa\*` into `my-openm-plus-plus-dir\html\`
- open your favorite browser at `http://localhost:4040` and refresh (clear browser cache if required)

Note: UI is beta version and you need to stop `oms` web-service in order to update, add or remove model .sqlite db files.*

Setup Development Environment

OpenM++ Requirements

Your development and runtime environment must meet following:

- OS: 64 or 32 bits version of:
  - Linux (tested): Debian 10 and 11, MX Linux 19 and 21, Ubuntu 20.04, RedHat 8
  - Windows (tested): 10, 7, 2016
  - MacOS 10.15 Catalina and 11.1+ Big Sur, including new Apple Arm64 CPU (a.k.a. M1)

Note: It does work on most of latest Linux'es, any Windows 7+ or 2008R2+, 32 and 64 bits. We just not testing it regularly on every possible Windows / Linux version.

- Support of c++17:
  - g++ 8.3+
  - Visual Studio 2019 or 2017 (VS 2017 not supported but may work), including Community Edition
  - Xcode 11.2+
- (optional) if want to build omc (openM++ compiler) from sources:
  - bison 3.3+ and flex 2.6+
- (optional) it is recommended to have MPI installed on your local machine or in your HPC cluster:
  - Linux (tested): OpenMPI 1.6+
  - Windows (tested): Microsoft MPI v8+, expected to work starting from HPC Pack 2012 R2 MS-MPI Redistributable Package
  - expected to work: MPICH (MS-MPI is in fact MPICH redistributed by Microsoft)

Optional development tools:

- R 3.5+ (version 4 not tested)
- Go 1.10+, on Windows required MinGw for g++ compiler
- node.js LTS version

Check c++17 capabilities

Linux: To check g++ version type: `g++ --version`, expected output:

```
g++ (Debian 8.3.0-6) 8.3.0              # Debian 10 and MX Linux 19
g++ (Debian 10.2.1-6) 10.2.1 20210110      # Debian 11 and MX Linux 21
g++ (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0  # Ubuntu 20.04
g++ (GCC) 8.3.1 20191121 (Red Hat 8.3.1-5) # RedHat 8, Rocky Linux, AlmaLinux
```

MacOS: To check c++ version type: `clang --version` or `g++ --version`, expected output:

```
Apple clang version 11.0.0 (clang-1100.0.33.12)
```

MacOS: install command line developer tools, if not installed already by Xcode: `xcode-select --install`

Windows: Make sure you have Visual Studio 2019 or 2017 installed with latest update (VS 2017 is not supported but may work).

If you are using different c++ vendor, i.e. Intel c++ then compile and run following test:

```cpp
#include <iostream>
#include <map>
#include <string>
#include <iostream>

using namespace std;

int main(int argc, char** argv)
{
    const map<string, string> capitals {
        { "Poland", "Warsaw"},
        { "France", "Paris"},
        { "UK", "London"},
        { "Germany", "Berlin"}
    };

    // print Country: Capital
    for (const auto & [k,v] : capitals)
    {
        cout << k << ": " << v << "\n";
    }
    return 0;
}
```

Save above code as `h17.cpp`, compile and run it:

```
g++ -std=c++17 -o h17 h17.cpp
./h17
```

Expected output:

```
France: Paris
Germany: Berlin
Poland: Warsaw
UK: London
```

Bison and Flex

Optional: If you want to recompile omc (OpenM++ compiler) then you need bison version >= 3.3 and flex 2.6+ installed.

To check bison and flex version type following commands:

```
bison --version
flex --version
```

Expected output:

```
bison (GNU Bison) 3.3.2  # Debian 10 and MX Linux 19
bison (GNU Bison) 3.5.1  # Ubuntu 20.04
bison (GNU Bison) 3.7.5  # Debian 11 and MX Linux 21
bison (GNU Bison) 3.0.4  # RedHat 8: this version is too OLD
flex 2.6.4
```

RedHat 8

You need a newer version of bison if you want to rebuild openM++ compiler (omc). One way of doing it is to rebuild bison from sources:

```
curl -o bison-src.tar.gz https://ftp.gnu.org/gnu/bison/bison-3.7.5.tar.gz
tar -xzf bison-src.tar.gz
cd bison-3.7.5
./configure --prefix=${HOME}/bison
make
make install
```

In order to use a newer version of bison export it to your environment:

```
export PATH=${HOME}/bison/bin:${PATH}
export LDFLAGS="-L${HOME}/bison/lib ${LDFLAGS}"
```

To verify result do `bison --version`, expected output:

```
bison (GNU Bison) 3.7.5
```

Potential issue: If `make` fail with error about missing `makeinfo` then you may need to install it from official RedaHat PowerTools repository:

```
dnf install dnf-plugins-core
dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
dnf config-manager --set-enabled powertools
dnf install texinfo
```

Windows:

- download Windows version of bison and flex
- if your OpenM++ checkout folder is: `C:\SomeDir\` then unzip `win_flex_bison-2.5.24.zip` into `C:\SomeDir\bin\`

To check bison and flex version type following commands with current directory `C:\SomeDir\bin\`:

```
win_bison --version
win_flex --version
```

Expected output:

```
bison (GNU Bison) 3.7.4
flex 2.6.4
```

MacOS Bison:

Bison version included in MacOS `bison (GNU Bison) 2.3` released in 2006 and too old for openM++. You can install bison 3.8 from HomeBrew or from (MacPorts)[https://www.macports.org/]

MacOS Bison from HomeBrew:

- install HomeBrew from GUI terminal:

  ```
  /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
  ```

- install bison 3.8 using HomeBrew:

  ```
  brew install bison@3.8
  ```

- export bison, you may also want to add it into your .zprofile: if MacOS on Intel CPU:

  ```
  export PATH="/usr/local/opt/bison/bin:$PATH"
  export LDFLAGS="-L/usr/local/opt/bison/lib ${LDFLAGS}"
  ```

  if MacOS on Apple Arm64 CPU (a.k.a. M1):

  ```
  export PATH="/opt/homebrew/opt/bison/bin:${PATH}"
  export LDFLAGS="-L/opt/homebrew/opt/bison/lib ${LDFLAGS}"
  ```

- verify bison version

  ```
  bison --version
  ....
  bison (GNU Bison) 3.8.2
  ```

Install MPI

OpenM++ is using MPI to run the models on multiple computers in your network, in cloud or HPC cluster environment.

Linux: To check your MPI version:

```
[user@host ~]$ mpirun --version
mpirun (Open MPI) 1.10.7
```

You may need to load MPI module in your environment on RedHat:

```
module load mpi/openmpi-x86_64
mpirun --version
```

Windows: To check your MPI version:

```
C:\> mpiexec /?
Microsoft MPI Startup Program [Version 10.0.12498.5]
........
```

Windows: download and install Microsoft MPI SDK and MPI Redistributable.

Test MPI

You can test your MPI environment with following code:

```cpp
#include <mpi.h>
#include <iostream>
using namespace std;

int main(int argc, char **argv)
{
    int mpiCommSize;
    int mpiRank;
    int procNameLen;
    char procName[MPI_MAX_PROCESSOR_NAME];

    MPI_Init(&argc, &argv);

    MPI_Comm_size(MPI_COMM_WORLD, &mpiCommSize);
    MPI_Comm_rank(MPI_COMM_WORLD, &mpiRank);
    MPI_Get_processor_name(procName, &procNameLen);

    cout << "Process: " << mpiRank << " of " << mpiCommSize << " name: " << procName << endl;

    MPI_Finalize();
    return 0;
}
```

Save this code as `mhp.cpp`, compile and run it:

```
mpiCC -o mhp mhp.cpp
mpirun -n 4 mhp
```

Expected output is similar to:

```
Process: 0 of 4 name: omm.beyond2020.com
Process: 2 of 4 name: omm.beyond2020.com
Process: 1 of 4 name: omm.beyond2020.com
Process: 3 of 4 name: omm.beyond2020.com
```

Windows: To build MPI tests in Visual Studio:

- create C++ command-line project
- adjust following in project properties:
  - VC Directories -> Include Directories -> C:\Program Files\Microsoft MPI\Inc
  - VC Directories -> Library Directories -> C:\Program Files\Microsoft MPI\Lib\i386
  - Linker -> Input -> Additional Dependencies -> msmpi.lib
- build it and run under Visual Studio debugger

Please use `amd64` version of MS MPI libraries if you want to build 64bit version.

To run MPI test on Windows type following in your command-line prompt:

```
mpiexec -n 4 mhp.exe
```

Expected output is similar to:

```
Process: 3 of 4 name: anatolyw7-om.beyond2020.local
Process: 2 of 4 name: anatolyw7-om.beyond2020.local
Process: 0 of 4 name: anatolyw7-om.beyond2020.local
Process: 1 of 4 name: anatolyw7-om.beyond2020.local
```

Install R

Download and install R version 3.5+ (v4+ not tested):

- Windows: https://cran.r-project.org/bin/macosx/R-3.6.3.nn.pkg
- on Linux use your package manager, e.g.: `sudo yum install R`
- MacOS on Intel CPU: https://cran.r-project.org/bin/macosx/R-3.6.3.nn.pkg

It is recommended to use RStudio or RStudio Server for development.

Install Go

- **Windows:**
  - download Go from https://golang.org/ and install into any directory, e.g.: `C:\Program Files\go`
  - download MinGw from your preferable distribution, ex: https://nuwen.net/mingw.html and unpack into any directory: `C:\MinGW\`
  - create your Go working directory, e.g.: `C:\go_workspace\`
  - set your environment variables:

    ```
    set GOPATH=C:\go_workspace
    set PATH=%GOPATH%\bin;%PATH%
    cd %GOPATH%
    C:\MinGW\set_distro_paths.bat
    ```

It is recommended to use Visual Studio Code for development.

- **MacOS on Intel CPU:** download and install fresh Go version, for example: https://golang.org/dl/go1.16.3.darwin-amd64.pkg
- **MacOS on Arm64 CPU:** download and install fresh Go version, for example: https://golang.org/dl/go1.16.3.darwin-arm64.pkg
- **MacOS** Go also can be installed from `go1.16.3.linux-amd64.tar.gz` or `go1.16.3.linux-arm64.tar.gz` archive, similar to Linux
- **MacOS**: include into your .zprofile PATH to Go, for example:

  ```
  export GOROOT=$HOME/go
  export PATH=$GOROOT/bin:${PATH}
  ```

  Note: above version number 1.16.3 is only an example, please most recent stable version.

- **Linux:**
  - download Go, for example version 1.16.3 from: https://golang.org/dl/go1.16.3.linux-amd64.tar.gz
  - unpack into any directory, e.g.: `~/go`
  - set your environment variables (in .profile or .bash_profile or .bashrc, etc.):

    ```
    export GOROOT=$HOME/go
    export PATH=$GOROOT/bin:${PATH}
    ```

If you want to copy models database content from SQLite to other vendors then you may also need to install unixODBC development package:

```
su -c "yum install unixODBC unixODBC-devel"
```

Currently supported database vendors are: SQLite (default), Microsoft SQL Server, MySql, PostgreSQL, IBM DB2, Oracle. You can use dbcopy utility to copy model data between any of vendors above, for example copy from MySQL to MSSQL or from PostgeSQL to SQLite.

Install node.js

You need `node.js` in order to build and develop openM++ UI. Please download and install stable version from Node.js.

Windows

- Use any of:
  - MSI installer: https://nodejs.org/dist/v14.16.1/node-v14.16.1-x64.msi
  - Zip archive: https://nodejs.org/dist/v14.16.1/node-v14.16.1-win-x64.zip
- if you are using archive then unpack it into `C:\node` directory and to start development open command prompt and type:

```
C:\node\nodevars.bat
cd C:\my-openm-plus-plus-dir\ompp-ui
npm install
```

Linux

- Use your favorite package manager
- Or directly download archive from Node.js and unpack into `$HOME/node`:

```
curl https://nodejs.org/dist/v14.16.1/node-v14.16.1-linux-x64.tar.xz -o node.tar.xz
mkdir $HOME/node
tar -xJf node.tar.xz -C node --strip-components=1
```

- add PATH to Node into your .bash_profile (or .profile or .bashrc, etc): `export PATH=$HOME/node/bin/:${PATH}`
- checkout and build UI:

```
cd my-openm-plus-plus-dir
git clone https://github.com/openmpp/UI.git ompp-ui
cd ompp-ui
npm install
npm run build
```

MacOS on Intel CPU

- Use any of:
  - Installer: https://nodejs.org/dist/v14.16.1/node-v14.16.1.pkg
  - Archive: https://nodejs.org/dist/v14.16.1/node-v14.16.1-darwin-x64.tar.gz
- if you are using archive then unpack it into `$HOME/node` and try checkout and build UI:

```
mkdir $HOME/node
tar -xzf node-v14.16.1-darwin-x64.tar.gz -C node --strip-components=1
```

- add PATH to Node into your .zprofile: `export PATH=$HOME/node/bin/:${PATH}`
- checkout and build UI as described in Linux section above

MacOS on Arm64 CPU

- install HomeBrew from GUI terminal:

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```

- install Node.js LTS version using HomeBrew:

```
brew install node@14
```

- add PATH to Node into your .zprofile: `export PATH=/opt/homebrew/opt/node@14/bin:${PATH}`
- checkout and build UI as described in Linux section above

Note: In examples above `node-v14.16.1` is an example of current LTS (long term support) version. Please check Node.js site to download latest LTS version.

2018, June: OpenM++ HPC cluster: Test Lab

Obsolete

HPC cluster Test Lab not available after October 2018. Instructions below outdated but may be useful as example of development test lab on Linux.

Where is OpenM++ and HPC cluster Test Lab

- Download: binary files
- Source code: openM++ git
- Documentation: this wiki
- HPC cluster (test lab): `ssh -p 4022 USER@omm.some-where.com`

OpenM++ HPC cluster test lab consists of:

- master node and 2 quad cores computational nodes.
- all nodes running 64bit Centos 7 and Open MPI.
- computational node names are: om1.some-where.com, om2.some-where.com
- shared directory to put executatbles: /mirror
- special user to run the tests on cluster: mpi
- script to run on cluster: /mirror/omrun
- cluster hosts description: /mirror/mpihosts

Please read Quick Start for OpenM++ Developers first. Additional information can be found in Linux section of Setup Development Environment.

Login to OpenM++ HPC cluster

To login on OpenM++ test lab cluster:

```
ssh -p 4022 USER@omm.some-where.com
```

If you are on Windows and using putty, please put following setting here:

```
server name: omm.some-where.com
port: 4022
Window -> Translation -> Remote Charter Set: UTF-8
```

Check your Environment

To verify OpenMPI is working:

```
 module load mpi/openmpi-x86_64
 mpirun -H omm,om1,om2 uname -n
```

expected output:

```
 omm.some-where.com
 om1.some-where.com
 om2.some-where.com
```

To verify c++ and OpenMPI development environment compile MPI Hello, World:

```cpp
#include <iostream>
#include <mpi.h>

using namespace std;

int main(int argc, char ** argv)
{
    int mpiCommSize;
    int mpiRank;
    int procNameLen;
    char procName[MPI_MAX_PROCESSOR_NAME];

    MPI_Init(&argc, &argv);

    MPI_Comm_size(MPI_COMM_WORLD, &mpiCommSize);
    MPI_Comm_rank(MPI_COMM_WORLD, &mpiRank);
    MPI_Get_processor_name(procName, &procNameLen);

    cout << "Process: " << mpiRank << " of " << mpiCommSize << " name: " << procName << endl;

    MPI_Finalize();
    return 0;
}
```

```
mpiCC -o /mirror/mhw mhw.cpp
cd /mirror
mpirun -H omm,om1,om2 mhw
```

Setup Your Environment

It is convenient to customize .bashrc to setup your environment:

```bash
# .bashrc
#
# ....something already here....
#

# enable MPI
#
source /usr/share/Modules/init/bash
module load mpi/openmpi-x86_64
```

Tip: If you want to have full Linux GUI on master node then freeNX client can be a good choice and Eclipse or Netbeans are excellent IDE for c++ development.

Build and Run OpenM++

Check out and compile OpenM++:

```
git clone https://github.com/openmpp/main.git master
cd master/openm/
make OM_MSG_USE=MPI
cd ../models/
make OM_MSG_USE=MPI all publish run
```

Copy build results to /mirror shared directory:

```
cp bin/* /mirror
```

Run the models on cluster with different number of subsamples:

```
cd /mirror
mpirun -H omm,om1,om2 -n 4 modelOne -General.Subsamples 4
```

you will be prompted for mpi user password, expected output is similar to:

```
2013-10-24 12:38:41.0360 Model: modelOne
2013-10-24 12:38:41.0359 Model: modelOne
2013-10-24 12:38:41.0360 Model: modelOne
2013-10-24 12:38:41.0363 Model: modelOne
2013-10-24 12:38:42.0518 Subsample 1
2013-10-24 12:38:42.0518 Subsample 2
2013-10-24 12:38:42.0520 Subsample 3
2013-10-24 12:38:43.0035 Subsample 0
2013-10-24 12:38:43.0062 Reading Parameters
2013-10-24 12:38:43.0062 Reading Parameters
2013-10-24 12:38:43.0062 Reading Parameters
2013-10-24 12:38:43.0063 Reading Parameters
2013-10-24 12:38:43.0066 Running Simulation
2013-10-24 12:38:43.0066 Writing Output Tables
2013-10-24 12:38:43.0066 Running Simulation
2013-10-24 12:38:43.0066 Writing Output Tables
2013-10-24 12:38:43.0066 Running Simulation
2013-10-24 12:38:43.0066 Writing Output Tables
2013-10-24 12:38:43.0066 Running Simulation
2013-10-24 12:38:43.0066 Writing Output Tables
2013-10-24 12:38:44.0198 Done.
2013-10-24 12:38:44.0198 Done.
2013-10-24 12:38:44.0198 Done.
2013-10-24 12:38:44.0200 Done.
```

Development Notes: Defines, UTF-8, Databases, etc.

OpenM++ development notes

This page contains various notes **only for OpenM++ developers**. There is no useful information on that page for anyone else. It is a notes, they are not in any specific order and may not true. OK, you have been warned.

Git layout of main repository

OpenM++ consists of 6 source code repositories published at GitHub / openmpp. Core portion of openM++ located at GitHub / openmpp / main and has following structure:

- bin - used for OpenM++ compiled binaries and third party tools
- include - includes for public interfaces of compiler and libraries
  - libopenm - model runtime library public interface
  - omc - model compiler public interface
- licenses - third party lincences
- models - test models, for example:
  - NewCaseBased - simple test model
  - NewTimeBased - simple test model
  - modelOne - test model for runtime library, does not use OpenM++ compiler
- openm - OpenM++ core source code
  - libopenm - model runtime library (libopenm) and compiler library (libopenm_omc_db)
    - common - common helper routines, for example: log
    - db - data access classes
    - include - includes for libopenm and libopenm_omc_db
    - model - model base classes
    - msg - message passing library
  - main.cpp - models main() entry point
  - libsqlite - SQLite with extension functions such as SQRT()
  - omc - OpenM++ compiler
- Perl - perl scripts
- props - VC++ project includes to build the models
- R - openMpp R library: integration between OpenM++ and R
- sql - sql scripts to create openM++ database
  - db2 - DB2 version of openM++ database scripts
  - mssql - Microsoft SQL Server version of openM++ database scripts
  - mysql - MySql version of openM++ database scripts
  - postgresql - PostgreSql version of openM++ database scripts
  - sqlite - SQLite version of openM++ database scripts

OpenM++ logs and trace

As it is now model executable output log messages into three streams:

- standard output (console)

- "last" log file: /current/working/dir/modelExeName.log
- "stamped" log file: /current/working/dir/modelExeName.date_time.pid.log

Model trace output is similar to log output but works much faster. Trace output is buffered and may be lost if something goes wrong and model crushed.

You can adjust output log and trace output inside of main() by changing: `theLog->init(...);` parameters. It is also be controlled by .ini options file.

Defines for OpenM++

You may need to change defines to build OpenM++ from source code:

- OM_DB_SQLITE: use SQLite as database provider (only one supported at the moment)
- OM_MSG_MPI: use MPI as for message passing library (see below)
- OM_MSG_EMPTY: use empty version message passing library (default value)
- OM_UCVT_MSSTL: use c++11 STL to convert strings to UTF-8 (default on Windows)
- OM_UCVT_ICONV: use glibc iconv to convert strings and file content to UTF-8 (default on Linux)

Please note:

- OM_MSG_MPI and OM_MSG_EMPTY mutually exclusive
- to set defines properly change `openm.build.props` (on Windows) or use `make OM_MSG_USE=MPI` (on Linux)
- OM_UCVT_MSSTL and OM_UCVT_ICONV mutually exclusive
- OM_UCVT_MSSTL tested on Windows with VC++2012 and account for Microsoft-specific implementation of STL `codecvt` classes.

Defines and other changes for VC++

Defines to compile libsqlite library with extension functions: SQLITE_ENABLE_COLUMN_METADATA; SQLITE_OMIT_LOAD_EXTENSION; HAVE_ACOSH; HAVE_ASINH; HAVE_ATANH;

To avoid innumerous compatibility errors and warnings following must be defined: _CRT_SECURE_NO_WARNINGS and _CRT_NONSTDC_NO_WARNINGS.

OpenM++ data library notes

IDbExec interface is db-connection wrapper and only the place where real SQL operations executed. All other classes are to wrap OpenM++ database tables and implement "business logic".

Data library is NOT thread-safe by design, do not pass it objects between model threads without proper guards.

Difference between OpenM++ database schema and Modgen schema:

- support multiple models and multiple versions of the same model
- support multiple run results of each model
- tends to be more "relational", i.e.:
  - language-specific rows moved to separate tables
  - sub-samples are in rows not in columns

Database schema "read-only" compatible with Modgen database. For each Modgen table corresponding view created which allow to read from OpenM++ database as from Modgen database. If OpenM++ database contains multiple models (or versions) then it not be exposed to Modgen compatibility views.

OpenM++ database notes

If database connection string is not specified then model try to open SQLite database with name ModelName.sqlite (i.e.: modelOne.sqlite) in current working directory. Other word, default database connection strig is:

```
Database=ModelName.sqlite; Timeout=86400; OpenMode=ReadWrite;
```

Database can be created by following commands:

```
cd
sqlite3 ModelName.sqlite < ../sql/sqlite/create_db_sqlite.sql
sqlite3 ModelName.sqlite < ModelName_create_model.sql
sqlite3 ModelName.sqlite < ModelName_insert_parameters.sql
```

On Linux slite3 executable most likely in your PATH. On Windows you must download sqlite3.exe from SQLite web-site.

OpenM++ data library notes: SQLite

Following parameters allowed for SQLite database connection:

- Database - (required) database file name or URI, file name can be empty
- Timeout - (optional) table lock "busy" timeout in seconds, default=0
- OpenMode - (optional) database file open mode: ReadOnly, ReadWrite, Create, default=ReadOnly
- DeleteExisting - (optional) if true then delete existing database file, default: false

If OpenMode=Create specified then database file created if not exist, which is default SQLite behavior.

Note: minimal connection string syntax for SQLite provider is: "Database=" and in that case SQLite will open temporary database. That kind of connection string does not really make sense for OpenM++ models because temporary database will be deleted after model exit.

OpenM++ message passing library notes

Message passing library (a.k.a. execute library) used for:

- broadcast metadata and input parameters from root process to slave modeling processes
- gather output modeling results from all modeling processes into root process

That library has two versions:

- define OM_MSG_MPI: MPI-based version which does the job as described above (MPI component must be installed)
- define OM_MSG_EMPTY: empty version of library, which does nothing and don't required anything installed

When empty version of library can useful?

To develop and debug your model without having MPI installed and without complexity of multi-process debugging. Obviously, some technique must be used to debug modeling logic inside of single process.

IMsgExec interface is main class for message passing library. All processes involved in the modeling must can be identified by integer process rank. Root process rank is zero.

Messaging library is NOT thread-safe, at least for now, do not pass it objects between model threads without proper guards. It may change in the future versions.

OpenM++ and UTF-8 strings

All strings inside of openM++ source code expected to be are UTF-8 encoded. If you need to pass string to openM++ API, please convert it to UTF-8 first. There is helper function which return file content converted as UTF-8 string:

```
string fileContent = fileToUtf8("someFile.txt");
```

Following rules applied to detect file encoding:

- if byte order mark (BOM) present in the file then it converted according to BOM
- if first 2048000 bytes of file are UTF-8 then file considered as UTF-8 and not converted
- if code page (encoding name) specified, i.e.: "English_US.1252" then it used for conversion

- default user code page (encoding name) used to convert file content to UTF-8

You can use optional parameter to explicitly specify code page (encoding name):

```
string fileContent = fileToUtf8("someFile.txt", "English_Canada.1252"); // Windows: CP-1252
string fileContent = fileToUtf8("someFile.txt", "WINDOWS-1252");        // Linux:   CP-1252
```

Encoding name is OS-specific and conversion would fail if name is invalid.

Note: conversion from UTF-32 to UTF-8 not supported on Windows.

Model digest, parameter digest, output table digest, etc.

OpenM++ is using MD5 digest to compare and find models, parameters, output tables and types in database. There are two digests calculated for model run:

- model run values digest which based on
  - values in model run output tables
  - values of model run input parameters
- model run metadata digest which is unique key of model run Model run values digest calculated only after run is completed. It can be empty if run failed.

Model run results do include output table values and all input parameter values. Model runs are stored in database as single copy only. For example, if digest of (parameter A value of model run 101) == digest of (parameter A value of model run 123) then only value from run 101 actually stored in database and run 123 is a link to run 101 value.

Following rules are used to calculate digests:

```
Model digest:
-------------
model name, model type, model version
for all model types:
  type digest
for all model parameters:
  parameter digest
for all model output tables:
  table digest

Parameter digest:
----------------
parameter name, rank, type digest
for all dimensions:
  id, name, size, type digest

Output table digest:
--------------------
table name, rank
for all dimensions:
  id, name, size (including "total" item), type digest
for all accumulators:
  acc id, name, source
  examples:
    id:     1
      name:   acc1
      source: accumulator 1: sum(delta(interval(duration(smoking_status, NON_SMOKER))))
    id:     9
      name:   Expr4
      source: 1.0E2 * ( acc4 / acc0 )
for all expressions (a.k.a. measures):
  id, name, source
  examples:
    id:     0
      name:   Expr0
      source: ( OM_AVG( acc0 ) / ( OM_AVG( acc1 ) - OM_AVG( acc2 ) ) )
    id:     8
      name:   E8
      source: OM_AVG(acc8)

Type digest:
------------
type name, dictionary id (e.g.: 3=range), "total" enum id
for all enums:
   id, enum name

Import digest for parameter or output table:
--------------------------------------------
```

```
rank, type digest
for all dimensions:
  id, name, size, type digest

Model run metadata digest:
--------------------------
model digest, run name, sub-values count, create date-time, run stamp

Model run value digest:
-----------------------
sub-values count, completed sub-values count, run status

for all parameters:
  parameter value digest

for all output tables:
  output table value digest

Value digest for parameters:
----------------------------
parameter_name, parameter_digest
sub_id, dimension names, param_value as comma separated header
  example (2 dimensions):
    sub_id,dim0,dim1,param_value
for all value rows:
  select sub_id, dimensions id, param_value
  convert sub_id, dimensions id into strings
  convert param_value to string
    if type is float then format as %.15g
    if type is boolean then "true" or "false"
  example (2 dimensions boolean):
    2,11,22,true

Value digest for output table:
------------------------------
table_name, table_digest

for all accumulators:
  accumulators value digest

for all expressions:
  expressions value digest

Value digest for output table accumulators:
-------------------------------------------
comma separated header: acc_id, sub_id, dimension names, acc_value
  example (2 dimensions):
    acc_id,sub_id,dim0,dim1,acc_value

for all value rows:
  select acc_id, sub_id, dimensions id, acc_value
  convert acc_id, sub_id, dimensions id into strings
  format acc_value as %.15g
  example (2 dimensions):
    2,15,11,22,0.1234

Value digest for output table expressions:
------------------------------------------
comma separated header: expr_id, dimension names, expr_value
  example (4 dimensions):
    expr_id,dim0,dim1,dim2,dim3,expr_value

for all value rows:
  select expr_id, sub_id, dimensions id, expr_value
  convert expr_id, sub_id, dimensions id into strings
  format expr_value as %.15g
  example (4 dimensions):
    1,11,22,33,44,0.789
```

2012, December: OpenM++ Design

About this document

This roadmap and architecture document presented from "model developer" point of view, which imply C++ development process, user aspects of OpenM++ are deliberately excluded. Please refer to OpenM++ user guide pages for additional details.

What is OpenM++

OpenM++ is an open source implementation of the Modgen microsimulation tool created at Statistics Canada. It is not a copy of the Modgen, but a new, functionally equal implementation of publically available Modgen specifications. OpenM++ also has its own important distinct features like portability, scalability and open source, which Modgen does not. Extensive information on Modgen is available on the Statistics Canada web site at http://www.statcan.gc.ca/microsimulation/modgen/modgen-eng.htm.

OpenM++ Design Basics

Common OpenM++ design principles:

- portability: it must work on Windows and Linux, 32 and 64 bit versions
- scalability: work on single PC, in cluster or in cloud environment
- open source: it is open source product

OpenM++ is portable and scalable:

OpenM++ designed, developed and tested to work on Windows and Linux, in 32 and 64 bits. As result same model can be created and tested on model developer Windows PC and later run on Linux (or Windows) HPC cluster with thousands CPUs.

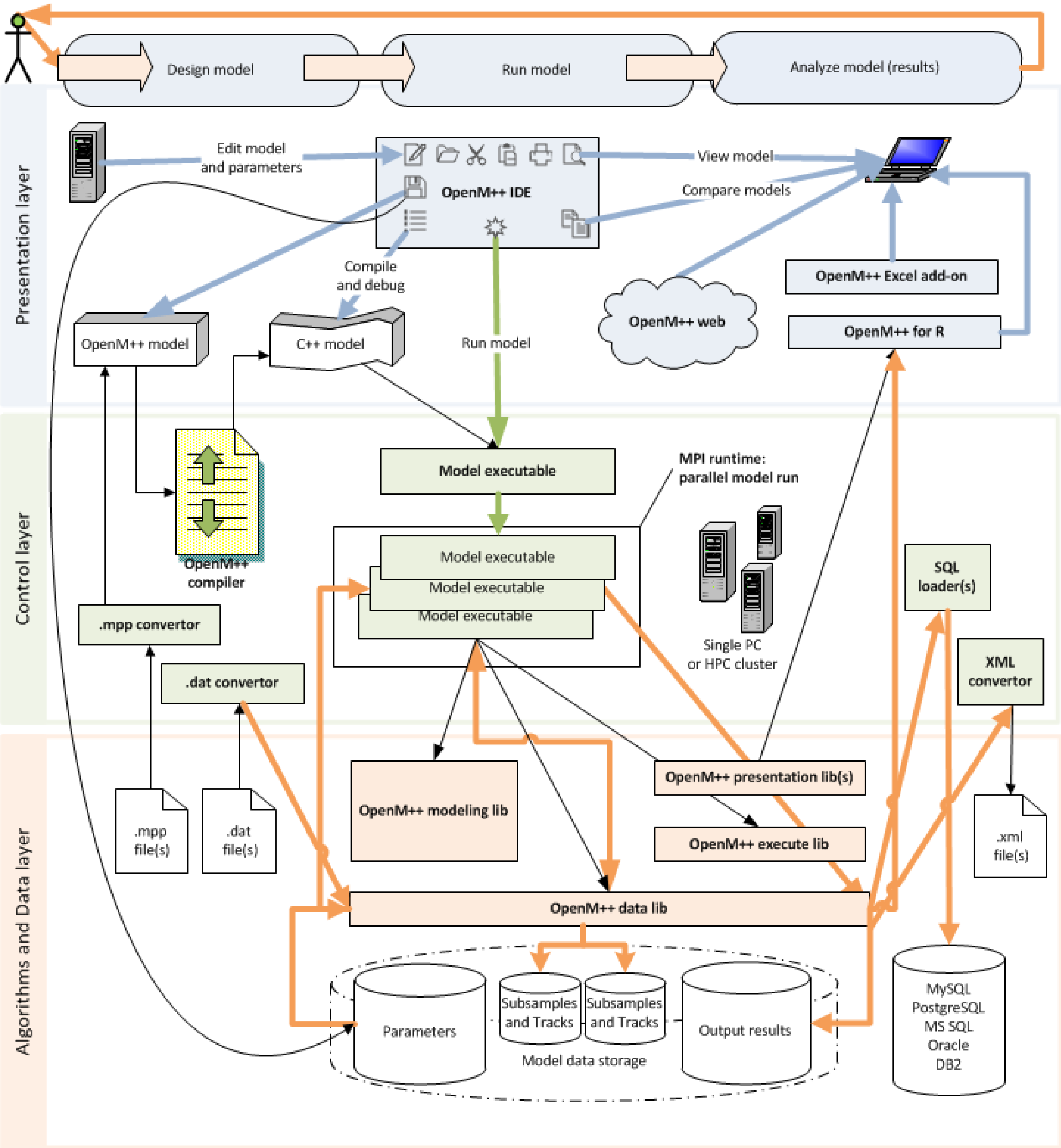
OpenM++ models are essentially highly parallelizable computational applications and fits very well in HPC cluster environment.

Specific version of cluster environment is to be established during first development phase. However, for the purpose of this design document, we can make a safe assumption cluster environment mean MPI cluster since many of existing HPC clusters, including CopmuteCanada cluster, are MPI-based.

OpenM++ is web-ready and cloud-ready:

It is important to understand, OpenM++ is targeted to provide “software-as-a-service” cloud models for research community. To simplify this roadmap cloud and web details of OpenM++ omitted here. However, OpenM++ cloud capabilities are essential and all control layer, algorithms and data layer components must be designed, developed and tested as cloud-ready.

OpenM++ Architecture



OpenM++ consists of 3 software layers:

- layer 1: presentation
- layer 2: control
- layer 3: algorithms and data and must accommodate to 3 model life-cycle stages:
- model design and development stage
- model run stage
- modeling results analysis stage

Note: Components described below in the order of OpenM++ layers and not in the order of development. For each component priority of the features specified as (pri1), (pri2) and (pri3); that value does NOT correspond to OpenM++ development phases.

Layer 1: OpenM++ presentation layer

Component 1.1: OpenM++ IDE

OpenM++ IDE is desktop GUI application to:

- (pri1) edit model parameters
- (pri1) view model output results

- (pri1) compare parameters of two models (see note below)
- (pri2) edit model source file(s) with (p3) syntax highlighting for OpenM++ language (.ompp)
- (pri2) compile from .ompp into c++ by invoking OpenM++ compiler, capture error s and warnings
- (pri2) compile and debug c++ model code by using GCC or Microsoft c++
- (pri2) debug c++ model executable
- (pri2) run model on single PC or (p3) submit it to HPC cluster
- (pri3) support source control system(s) integration (svn and/or git)
- (pri2) provide unit testing functionality

Note1: As an alternative OpenM++ GUI can be split into multiple independent applications with desktop or web UI. In any case it must provide and parameter editing capabilities.

Note2: Model comparison initially implemented as simple tool to compare parameters of two models. It can be later extended to support output results comparison with sophisticated analysis, however, most likely it going to be done as part of described below OpenM++ model analysis tools and OpenM++ web solutions.

Component 1.2: OpenM++ output result viewers and model analysis tools

OpenM++ presentation layer should be extendable and must support development of 3rd-party tools to view and analyze model output results. Following viewers to be implemented first:

- (pri1) Excel workbook and /or sample module(s)
- (pri2) import/export into R
- (pri2) basic web UI sample pages for ASP.NET
- (pri3) basic web UI sample pages for PHP
- (pri3) basic web UI sample pages for Java
- (pri3) Excel OpenM++ add-on

Basic web UI sample pages with necessary server-side components provided as reference point for web development and should allow view/edit parameters, view output results and run model executable.

Component 1.3: OpenM++ cloud and web capabilities

OpenM++ must be cloud-ready and support “software-as-a-service” usage of the model(s). These capabilities are out of current document scope. Mentioned above OpenM++ basic web UI sample pages provide starting point for web-developers. As next step web-solutions to use OpenM++ models on the web are going to be developed:

- (pri1) OpenM++ ASP.NET web solution (comparable to ModgenWeb)
- (pri2) OpenM++ PHP web solution
- (pri3) OpenM++ Java web solution

Those web-solutions (as every other portion of OpenM++) must be scalable and portable and ready to be deployed in private or public cloud platform, for example, Microsoft Azure for OpenM++ ASP.NET web solution (specific cloud platforms to be established). Based on that OpenM++ cloud software service capabilities can be created to provide ability for researches to work with their own models, including collaborative modeling, by using thin clients (i.e. web-browsers).

Note: Full C++ model development cycle is not supported by web solutions, however it may be considered as OpneM++ cloud the feature.

Layer 2: OpenM++ controller layer

That layer should provide set of command-line utilities, scripts and components to:

- compile, debug and run OpenM++ models on single PC or in cluster environment
- import, export and convert model data

Component 2.1: OpenM++ compiler

(pri1) The OpenM++ compiler produces C++ code from .ompp source code for a specific model. The .ompp source code is written by a model developer and contains declarative and procedural elements. Declarative elements include types, parameters, agents, variables, inter-agent links, agent collections, events, and cross-tabulations. Procedural elements include code to implement events and (optionally) to prepare secondary inputs and outputs. The OpenM++ compiler also produces a database of meta information on the model which is used by other OpenM++ components.

Component 2.2: OpenM++ controller for MPI cluster

OpenM++ models should run in not only on single PC but also in HPC cluster. OpenM++ cluster controller is a command-line utility, script or set of scripts to support most commonly used HPC cluster environments. For the purpose of this document MPI-compatible environment is assumed, however, other options can be considered as well. Following steps required in order to implement this:

- (pri1) organize test OpenMPI or MPICH2 cluster for CentOS 64bit
- (pri1) establish development environment for Windows 32bit and 64bit
- (pri1) create OpenM++ controller(s) for each cluster environment
- (pri2) establish automated test procedures for OpenM++ models in cluster
- (pri3) organize test OpenMPI or MPICH2 cluster for Debian or Ubuntu 64bit
- (pri3) organize test MS HPC cluster for Windows 64bit

Component 2.3: Modgen compatibility convertors

These are a command-line utilities to convert existing Modgen models into OpenM++ format:

- (pri1) parameters .dat file(s)
- (pri2) source model code .mpp file(s)

Component 2.4: OpenM++ SQL loaders

This is a command-line utility(s) to load data from OpenM++ model data storage into well-known SQL Server databases:

- (pri1) loader for MS SQL Server
- (pri1) loader for MySQL / MariaDB
- (pri2) generic SQL99 loader
- (pri3) loader for Oracle
- (pri3) loader for PostgreSQL
- (pri3) loader for IBM DB2
- (pri3) loader for Apache Derby, H2 or HSQL
- (pri3) loader for LucidDB, InfiniDB or MonetDB

Note: As an alternative solution all or some above functionality can be moved into OpenM++ data library. It also possible to supply few different versions of OpenM++ data library targeted to the different SQL Server database.

Component 2.5: OpenM++ output convertors

This is a command-line utility(s) to convert from OpenM++ model data storage into well-known formats:

- (pri1) .csv convertor for parameters and output results
- (pri2) .xml convertor for model data or user-defined subset of model data
- (pri3) SDMX convertor for model data
- (pri3) convertor into Statistics Canada Biobrowser database

Layer 3: OpenM++ algorithms and data layer

This layer consists of OpenM++ common libraries and model data storage (model database).

Component 3.1: OpenM++ modeling library

The modeling library provides core functionality for the model life cycle, including agent creation / destruction, event queue management, on-the-fly cross-tabulation, and pre- and post-simulation processing. It may use OpenM++ data and execute libraries to organize model execution and result aggregation (especially in cluster environment), read model parameters, save model tracks and aggregate cross-tabulation results.

Component 3.2: OpenM++ model data storage (model database)

OpenM++ data storage design should provide an ability to store model parameters and output results inside of SQL database and support model tracking functionality, which may be done through a different database, text or XML file (subject for research during phase 1). OpenM++ data storage can be implemented in following ways:

- (pri1) inside of single embedded (file-based) SQL database
- (pri2) as above plus extra database for model tracking
- (pri3) model parameters and metadata inside of file-based SQL database and output results as .csv files
- (pri3) inside of SQL server database chosen by model developer (i.e. MSSQL, Oracle, etc.)

In any case model data storage should support basic OpenM++ design principles:

- portability between Linux, Windows, 64 and 32bit OS's
- scalability from single PC up to HPC cluster environment

Component 3.3: OpenM++ data library

Data library(s) is a C++ library to support model data read/write operations and hide low-level implementation details to simplify model code and modeling library. As (priority 1) it should support single embedded (file-based) SQL database in portable way. However, in a future (priority 3) it can consist of different implementations of data libraries for different target model storage (for example, to directly write into Oracle).

(priority 2) Second part of OpenM++ data libraries should provide an access to model data from Java and .NET to allow develop model analyzing tools and OpenM++ web solutions.

Component 3.4: OpenM++ execution library

(pri1) Execution is relatively thin C++ layer to simplify modeling library scalable coding, or other words, to avoid low-level details inside of modeling library for handling the difference between single PC and cluster execution. Depending on design decisions and target cluster environment it may not be used directly from modeling library but rather called from OpenM++ cluster controllers (see 2.2). In any case it should:

- (pri1) provide necessary information for model initialization (i.e. number of CPUs)
- (pri1) synchronize parallel model execution (i.e. wait for completion)
- (pri2) support data exchange between models or model and controller (i.e. progress report)
- (pri2) simplify tracking data exchange
- (pri1) organize transparent communication for output result aggregation

For the purpose of this document MPI cluster environment assumed, however other options can be considered as well.

(pri1) It is important to understand the modeling library may be designed in “single-threaded” way and then execution library must organize additional thread(s) for the purpose of model cluster communication, progress reporting, tracking, etc. Multithreading must be done in portable way and following solution should be considered for research during phase 1 of development:

- STL and C++11 standard features for threading and synchronization (i.e.: future)
- glib
- boost::thread and synchronization libraries
- APR (Apache portable runtime)

- OpenMP

Component 3.5: OpenM++ presentation library(s)

(pri1, pri2, pri3) Presentation libraries together with data library allow developing applications to view and analyze OpenM++ model output results. Priority and functionality of presentation libraries development completely defined by priority of OpenM++ viewers, and OpenM++ web solutions, described in 1.2 and 1.3 above. As (pri1) priority .NET presentation library(s) for Excel viewer and ASP.NET basic UI should be implemented.

2012, December: OpenM++ Model Architecture, December 2012

About this document

This roadmap and architecture document presented from "model developer" point of view, which imply C++ development process, user aspects of OpenM++ are deliberately excluded. Please refer to OpenM++ user guide pages for additional details.

OpenM++ model use cases

OpenM++ by design is portable and scalable environment which allow researchers to run same model on single Windows PC and on Linux (or Windows) HPC cluster by simply re-compiling model C++ code for target platform. For example, model developer can use Visual Studio on his own Windows PC to write, test and debug the model and later send model .cpp code to other researcher who can build and run that model on Linux HPC cluster with hundreds CPUs.

There are four main groups of openM++ model users:

- developer: using C++ IDE with openM++ installed to develop and run models mostly on their local PC
- researcher: uses openM++ models created by developer executable to run simulation on local workstation and/or on HPC cluster
- institutional user: member of research organization with advanced IT infrastructure who mostly running openM++ models in resource-shared environment (i.e. over the web)
- public user: member of the general public using simplified interface over the web.

Those user groups do have distinctive hardware / software environments and different requirements to model architecture:

- developer:
  - mostly local Windows or Linux PC with GUI
  - run the model hundred times to debug it
  - have full admin privileges on his local machine
  - eventually need to pack model executable and data files and send it to researcher
- researcher:
  - HPC cluster (large or small) or local Windows, Linux without GUI
  - run the model multiple times and collect the results
  - run the model 100's or 1000's of times for Probabilitistic Sensitivity Analysis or for model estimation.
  - do not have admin privileges, especially on cluster
  - often need to pack model data files to publish it, move from local PC to HPC cluster or share with other researchers
- institutional user:
  - uses web UI to run the model in cloud, on HPC cluster or other powerful server environment
  - have absolutely no access to actual server environment
  - at any time can use IT department to deploy openM++ models in cloud, create modeling web-sites, manage model database on SQL server, etc.
- public user:
  - runs a version of a model via the web written and compiled in openM++ with a limited set of parameters and limited set of output screens, possibly in parallel with hundreds of other general public users.
  - very limited if any capacity at all to save results between sessions.

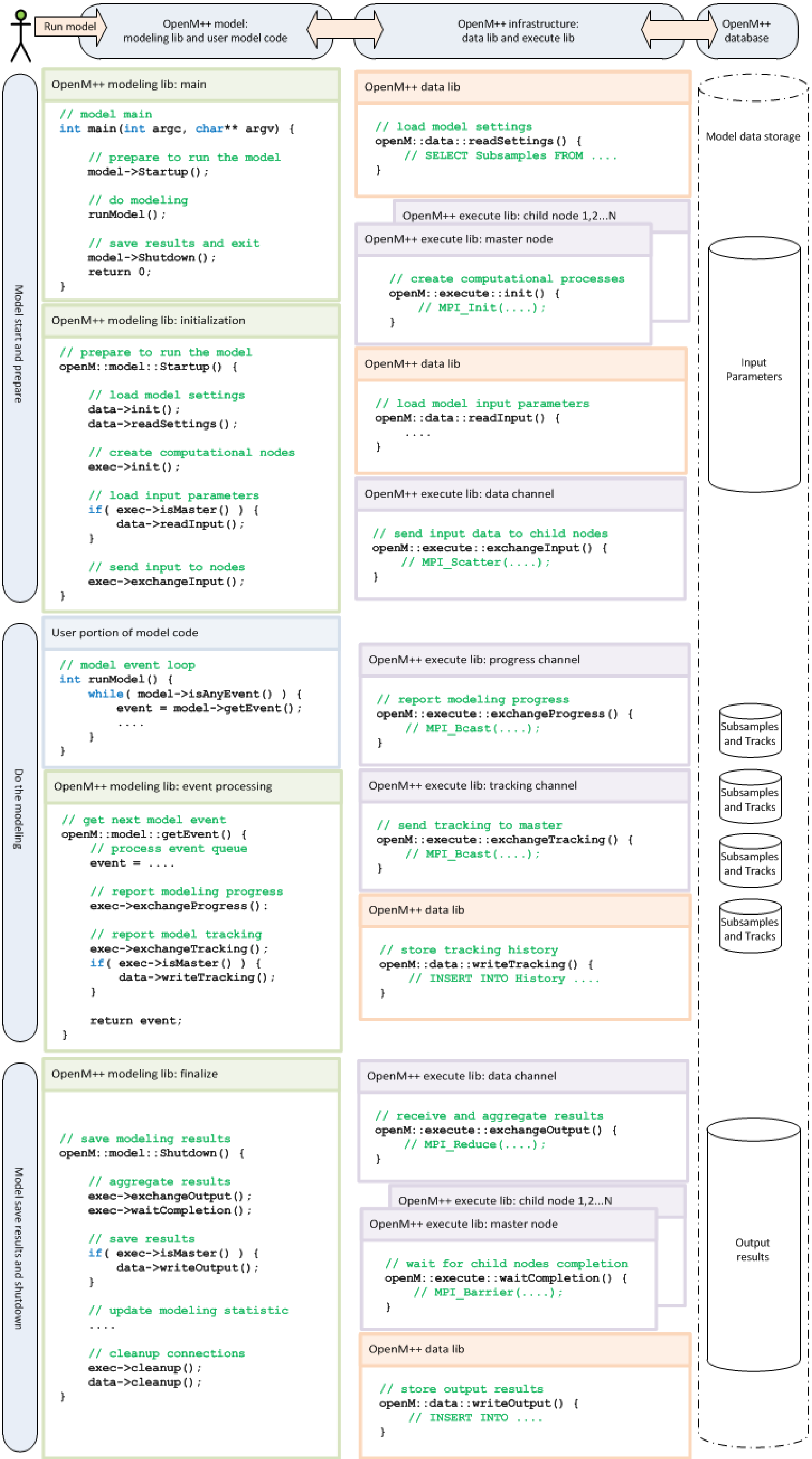
It is typical for openM++ users to not have advanced IT management skill as they are highly regarded professionals in their own area of interest. It may also not always possible for openM++ user to install additional software in their environment (i.e. in public HPC cluster). From that point easiest way of model deployment and model data export-import can be done through simple file operations (file copy). It is obviously not suitable for institutional users, however they can: (a) rely on dedicated IT department resources if necessary and (b) do have installed and supported web-servers, SQL databases servers and other resources where openM++ cloud components can be deployed.

Based on those use cases openM++ model architecture assumes following preferences:

- model, input parameters and output results available as set of files
- user may not want to (or can't install) database client-server software to store model data

Note: To simplify description of model architecture below it is done from developer or researcher user point of view and web cloud aspects are deliberately excluded.

OpenM++ model run cycle



Run model
OpenM++ model:
modeling lib and user model code
OpenM++ infrastructure:
data lib and execute lib
OpenM++
database
Model start and prepare
OpenM++ modeling lib: main
// model main
int main(int argc, char** argv) {
// prepare to run the model
model->Startup();
// do modeling
runModel();
// save results and exit
model->Shutdown();
return 0;
}
OpenM++ modeling lib: initialization
// prepare to run the model
openM::model::Startup() {
// load model settings
data->init();
data->readSettings();
// create computational nodes
exec->init();
// load input parameters
if(exec->isMaster()) {
data->readInput();
}
// send input to nodes
exec->exchangeInput();
}
OpenM++ data lib
// load model settings
openM::data::readSettings() {
// SELECT Subsamples FROM ....
}
OpenM++ execute lib: child node 1,2...N
OpenM++ execute lib: master node
// create computational processes
openM::execute::init() {
// MPI_Init(....);
}
OpenM++ data lib
// load model input parameters
openM::data::readInput() {
....
}
OpenM++ execute lib: data channel
// send input data to child nodes
openM::execute::exchangeInput() {
// MPI_Scatter(....);
}
Model data storage
Input
Parameters
Do the modeling
User portion of model code
// model event loop
int runModel() {
while(model->isAnyEvent()) {
event = model->getEvent();
....
}
}
OpenM++ modeling lib: event processing
// get next model event
openM::model::getEvent() {
// process event queue
event = ...
// report modeling progress
exec->exchangeProgress():
// report model tracking
exec->exchangeTracking();
if(exec->isMaster()) {
data->writeTracking();
}
return event;
}
OpenM++ execute lib: progress channel
// report modeling progress
openM::execute::exchangeProgress() {
// MPI_Bcast(....);
}
OpenM++ execute lib: tracking channel
// send tracking to master
openM::execute::exchangeTracking() {
// MPI_Bcast(....);
}
OpenM++ data lib
// store tracking history
openM::data::writeTracking() {
// INSERT INTO History ....
}
Subsamples and Tracks
Subsamples and Tracks
Subsamples and Tracks
Subsamples and Tracks
Model save results and shutdown
OpenM++ modeling lib: finalize
// save modeling results
openM::model::Shutdown() {
// aggregate results
exec->exchangeOutput();
exec->waitCompletion();
// save results
if(exec->isMaster()) {
data->writeOutput();
}
// update modeling statistic
....
// cleanup connections
exec->cleanup();
data->cleanup();
}
OpenM++ execute lib: data channel
// receive and aggregate results
openM::execute::exchangeOutput() {
// MPI_Reduce(....);
}
OpenM++ execute lib: child node 1,2...N
OpenM++ execute lib: master node
// wait for child nodes completion
openM::execute::waitCompletion() {
// MPI_Barrier(....);
}
OpenM++ data lib
// store output results
openM::data::writeOutput() {
// INSERT INTO ....
}
Output
results

Because openM++ models can scale from single PC to HPC cluster model execution (model run-cycle) depends on environment.

Simple (single PC) case *(italic indicates optional)*:

- start of model executable (model.exe)
- read model settings from database (read execution scenario)
- read model input data from database
- run modeling loop:
  - execute user model code
  - *report on model progress if required*
- *do model results aggregation if required*
- write results into database output tables
- finally report execution statistics and exit

If model runs in cluster environment then openM++ can transparently create multiple copies of model executable process and distribute it on cluster nodes.

Model run-cycle on cluster *(italic indicates optional)*:

- start of master model executable (model.exe)
- read model settings from database (read execution scenario)
- detect run-time environment
- spawn model.exe processes on computational nodes
- read model input data from database
- distribute input data between all computational nodes
- run modeling loop:
  - execute user model code
  - *report on model progress if required*
  - *collect model tracking information to debug the model*
- wait until all modeling completed on all computational nodes
- collect model results from each node
- *do results aggregation if required*
- write results into database output tables
- finally report execution statistics and exit

Note: It is important to understand the diagram on that page represent schematic picture and real openM++ code may be significantly more complex. For example, report modeling progress call exchangeProgress() may not actually do anything but place a data in the buffer and separate thread would do actual master-slave communication and progress report.

OpenM++ modeling library

The modeling library provides core functionality for the model run-cycle as it is described above. It contains main() entry point, it does agent creation / destruction, event queue management, on-the-fly cross-tabulation, and pre- and post-simulation processing.

It uses OpenM++ data and execute libraries to organize model execution (especially in cluster environment), read model input parameters, save model tracks and aggregate cross-tabulation results:

- for each input parameter model library by known data type, shape and other necessary information (memory address if required) to instantiate class object and populate it with values by calling data library

- for each output table result model library call data library to save results in model data storage (model database)

OpenM++ model data storage (model database)

OpenM++ data storage should provide an ability to store model parameters and output results. It consist of model data storage (model database), data library and, optionally, can use execute library to organize communication between computational nodes.

It can be implemented in following ways:

- option 0. flat files: directly read-write into flat text (XML, CSV, etc.) files
- option a. flat files + buffering (or MPI-IO): use memory buffering (or MPI-IO) to organize large size chunks and reduce data exchange in cluster environment
- option b. client-server database: use MySQL or other open source SQL server database
- option c. file-based (embedded) SQL database: use file-based database (i.e. SQLite) inside of master process and write custom code to emulate client-server for computational nodes

Evaluating those options from point of view openM++ use cases described above:

Option 0: direct write to flat files may not be realistic approach in cluster environment because:

- computational nodes most likely don't have locale file system
- global shared file system may have very high or prohibitive cost for small write operations. For example, if 100 model executables from 100 computational nodes want write to 100 bytes it may be, in worst case, 100 times slower than if master node writes 100*100 bytes. Of course, MPI-IO can solve that problem.

Option a: flat files + buffering (or MPI-IO)

- pros:
  - most human readable format
  - no additional tools required to create or modify model data, it can be done by any text editor
  - minimal development efforts
- cons:
  - real model data input typically bigger than user can type-in and maintain without additional tools
  - to analyze the data in any other software (i.e. Excel, R, SAS) custom data converter(s) must be developed

Option b: client-server

- pros:
  - relatively easy to implement
  - good performance is almost guaranteed
  - hundreds tools to read, compare and manipulate the data
- cons:
  - require to install and administer SQL server database which many openM++ users, such as model developers and independent researchers may have no right to do or may not want to do

Option c: file-based database (i.e. SQLite)

- pros:
  - hundreds tools to read and manipulate the data (i.e. Firefox SQLite manager add-on)
  - relatively easy to transfer to any database or exchange the data between researchers
- cons:
  - development time to create client-server code for cluster environment much higher than any other options

- it is less convenient as flat text files

OpenM++ data storage roadmap:

OpenM++ data storage can be implemented in following order of priorities:

- (pri1) inside of single embedded (file-based) SQL database
- (pri2) as above plus extra database for model tracking
- (pri3) model parameters and metadata inside of file-based SQL database and output results as .csv files
- (pri3) inside of SQL server database chosen by model developer (i.e. MSSQL, Oracle, etc.)

OpenM++ data library

Data library(s) is a C++ library to support model data read/write operations and hide low-level implementation details to simplify model code and modeling library. It is important to understand there is no "one size fit all solution" and openM++ must provide multiple versions of data library for different target model storage. For example, for model developer SQLite data library may be most convenient, however when openM++ installed as part of web solution then MySQL data library suites more.

Following priority order of data libraries implementation planned:

- (pri1) SQLite as embedded (file-based) database
- (pri2) generic ODBC tested with MySQL (MariaDB), PostgreSQL, MS SQL, Oracle and IBM DB2
- (pri3) flat text files version of data library (using MPI-IO)
- (pri3) MySQL (MariaDB) native client (non-ODBC)
- (pri3) PostgreSQL native client (non-ODBC)

List above is not final and can be changed anytime. Many other options also considered for development of specialized data library version. For example, libmysqld, Firebird, MS Access reviewed as potential candidates for embedded (file-based) database. Also MPI-IO, HDF5, NetCDF considered as foundation for flat text files data library version. And in the future releases it is very much possible to have native client (not ODBC-based) version of data library for MS SQL, Oracle and IBM DB2.

Keep in mind data library is part of the model run-time and not be ideal choice for other purpose. Most easy way to integrate openM++ with existing products is to use SQL loaders or output convertors. It allows to import or export data from openM++ data storage into other well-known SQL servers, i.e. from SQLite into MS SQL or dump it into flat text files (i.e. CSV, XML).

2012, December: Roadmap, Phase 1

OpenM++ Roadmap (phase1)

OpenM++ design details, components and priorities are defined on OpenM++ design page. Due to research nature of the project OpenM++ components, specific technologies and sequence of development must be periodically reviewed and can be changed.

Following results expected to be delivered at the end of the phase1 project (enumeration corresponds to OpenM++ design):

- OpenM++ compiler (2.1 priority1)
- OpenM++ controller for MPI cluster (2.2 priority1)
- OpenM++ modelling library (3.1 priority1)
- OpenM++ model data storage design (3.2 priority1)
- OpenM++data library (3.3. priority1)
- OpenM++ execute library (3.4 priority1)

Items above should allow to:

- create simple OpenM++ model
- compile model
- run model on (3.4 priority1) platforms (Windows and Linux, 32 and 64 bit, single PC and cluster)
- read parameters from and write results into OpenM++ model data storage

If time and resources permits following items also going to be delivered as result of the project:

- OpenM++ result viewers for Excel (1.2 priority1)
- OpenM++ basic web UI sample pages for ASP.NET (1.2 priority2)
- OpenM++ presenation libraries for .NET (3.5 priority1)
- compatibility convertor for Modgen parameters .dat files (2.3 priority1)
- compatibility convertor for Modgen source model code .mpp files (2.3 priority2)

Results of OpenM++ phase1 project effectively would cover:

- most existing Modgen desktop functionality, except of GUI
- ModgenWeb functionality on a prototype level (optional result)

Overall phase1 steps

1. Requirements and infrastructure stage (see step 1 below). **Time:** one calendar month
2. Compiler and runtime prototype stage (steps 2 and 3). **Time:** 2-3 months
3. Compiler and runtime alpha version stage (steps 4 and 5). **Time:** 4-6 months
4. Optional OpenM++ phase1 components (steps 8-11). **Time:** 6-16 weeks
5. OpenM++ public beta release stage (step 12). **Time:** 6-8 weeks

Total Time: one year, excluding optional steps

Detailed phase1 roadmap

1. Requirements, risks and technologies evaluation, tools, platforms and infrastructure setup

   Time: one calendar month

   Result: publically available design documents and development infrastructure

- Establish OpenM++ roadmap, licensing terms, evaluate targeted platforms (i.e. versions of Linux, cluster environments, etc.)
- Create OpenM++ controller for MPI cluster (2.2 priority1)
- Evaluate open source project hosting service and development tools required
- Create OpenM++ project by publishing roadmap and licence(s)

2. OpenM++ data storage design and libraries prototyping

   Time: 2-3 months (must be done together with step 3 below)

   Result: Prototype of OpenM++ compiler and runtime libraries

   - Prototype of OpenM++ modelling library (3.1 priority1) to be used by step 3
   - OpenM++ model data storage design (3.2 priority1)
   - Initial version of OpenM++data library (3.3. priority1)
   - Initial version of OpenM++execute library (3.4. priority1)

3. Initial version of OpenM++ modeling library

   Time: 2-3 months (must be done together with step 2 above)

   Result: Prototype of OpenM++ compiler and runtime libraries

   - Initial version of OpenM++ modelling library (3.1 priority1)
   - Initial version of OpenM++ compiler (2.1 priority1)

4. OpenM++ compiler and modeling library

   Time: 4-6 months?? (must be done together with step 5 below)

   Result: Alpha version of OpenM++ compiler and runtime libraries

   - First release of OpenM++ compiler (2.1 priority1), sufficient to compile simplest model
   - First release of OpenM++ modelling library (3.1 priority1)

5. OpenM++ execute and data libraries

   Time: 4-6 months?? (must be done together with step 4 above)

   Result: Alpha version of OpenM++ compiler and runtime libraries

   - First release of OpenM++ execute library (3.4. priority1)
   - First release of OpenM++data library (3.3. priority1)
   - First release of OpenM++ model data storage design (3.2 priority1)
   - First release of OpenM++ cluster controllers (2.2 priority1)

6. Results review, roadmap adjustment

   Time: one calendar week

   Result: Updated roadmap document and adjusted project plan

7. (optional) Initial version of OpenM++ presentation library(s) for .NET (3.5 priority1)

   Time: 2-4 weeks

   Result: Alpha version of OpenM++ presenation libarary for .NET

8. (optional, depends on step 7) OpenM++ for Excel (1.2 priority1)

   Time: 2-4 weeks

Result: Beta version of OpenM++ for Excel

  - First release of OpenM++ result viewers for Excel (1.2 priority1)
  - First release of OpenM++ presentation library for .NET (3.5 priority1) (it may be Excel-specific library)
  - OpenM++ compiler and runtime libraries bug fixes discovered during development

9. (optional, depends on step 7) OpenM++ basic web UI sample pages for ASP.NET (1.2 priority2)

   Time: 2-4 weeks

   Result: Beta version of OpenM++ web UI primer for ASP.NET

   - First release of OpenM++ basic web UI sample pages for ASP.NET (1.2 priority2)
   - First release of OpenM++ presentation library(s) for .NET (3.5 priority1) (this is may be ASP.NET specific)
   - OpenM++ compiler and runtime libraries bug fixes discovered during development

10. (optional, depends on step 7) First release of compatibility convertor for Modgen parameters .dat files (2.3 priority1)

    Time: 2-4 weeks

    Result: Beta version of OpenM++ convertor for Modgen parameters .dat files

11. (optional) First release of compatibility convertor for Modgen source model code .mpp files (2.3 priority2)

    Time: 2-4 weeks

    Result: Beta version of OpenM++ convertor for Modgen source code .mpp files

12. First public release

    Time: 6-8 weeks

    Result: Public beta version of OpenM++

    - Project documentation
    - Final testing and bug fixes
    - Project review and roadmap adjustment
    - First public release

2013, May: Prototype version

OpenM++ Prototype Version: May 2013

Initial openM++ prototype released on May 2013. It includes:

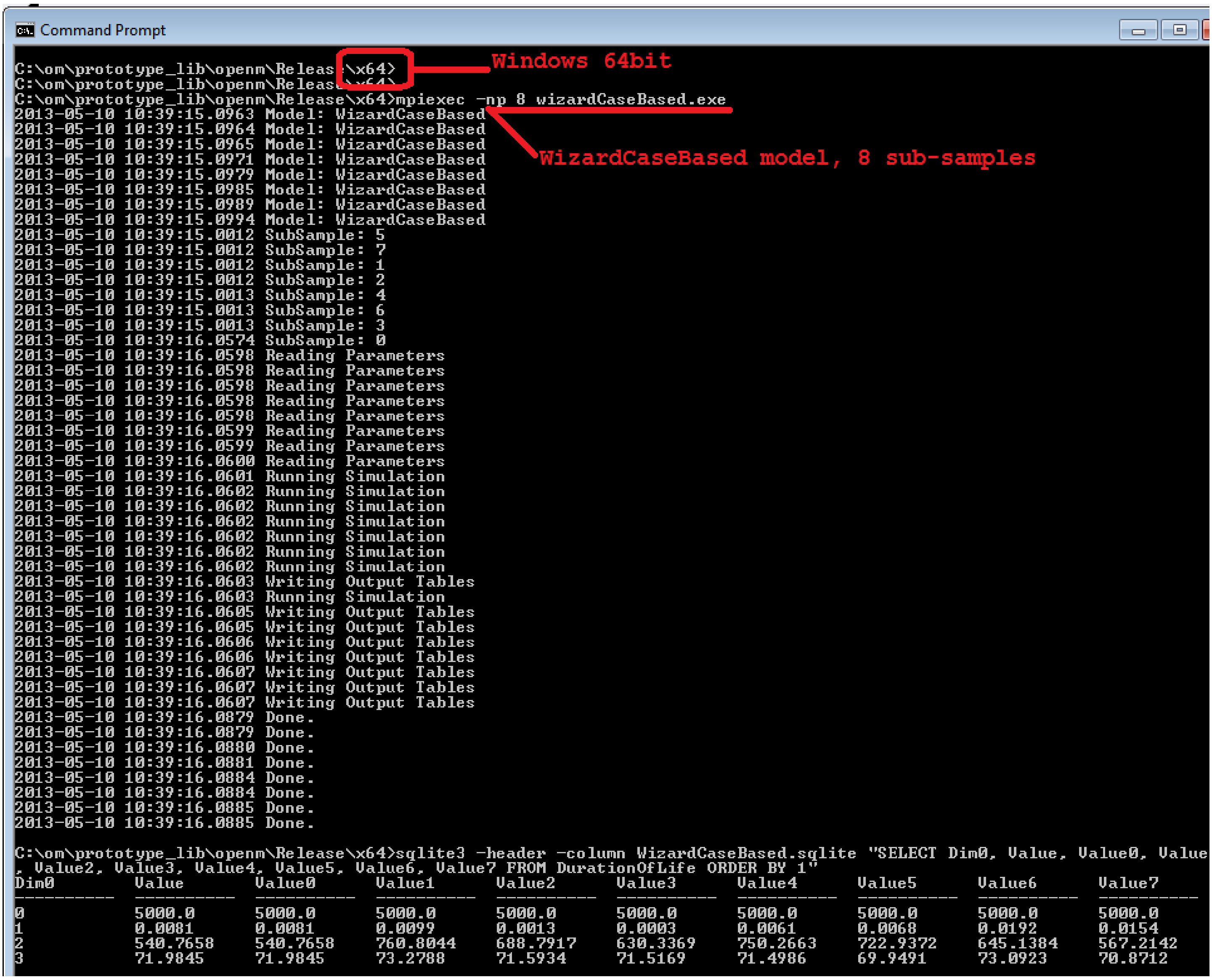
- openM++ compiler (initial prototype)
- runtime library (combined model, data and execute libs)
- two models, compiled, build and running on all target platforms

Important results are:

- openM++ is portable and highly scalable and can run single PC to supercomputing clusters
- openM++ model produce identical results for all platforms and matching existing Modgen results

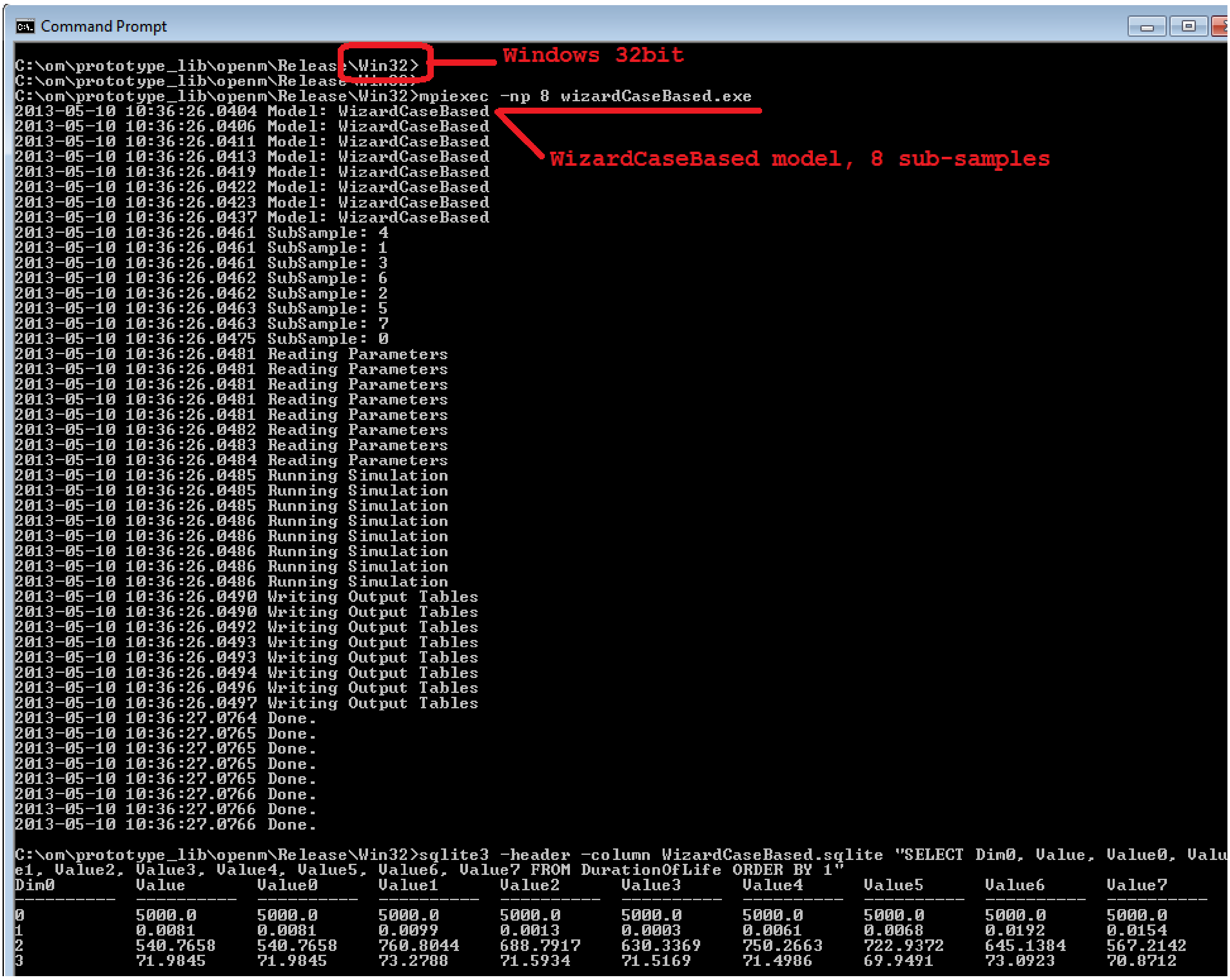
Below screenshots captured from openM++ WizardCaseBased model running on:

- Windows 64bit, 8 subsamples
- Windows 32bit, 8 subsamples
- Linux 64bit MPI cluster, 8 subsamples
- Linux 32bit non-clustered and without MPI, 1 subsample
- Linux 64bit HPC cluster at ComuputeCanada, 512 subsamples




```
C:\om\prototype_lib\openm\Release\x64>
C:\om\prototype_lib\openm\Release\x64>
C:\om\prototype_lib\openm\Release\x64>mpiexec -np 8 wizardCaseBased.exe
2013-05-10 10:39:15.0963 Model: WizardCaseBased
2013-05-10 10:39:15.0964 Model: WizardCaseBased
2013-05-10 10:39:15.0965 Model: WizardCaseBased
2013-05-10 10:39:15.0971 Model: WizardCaseBased
2013-05-10 10:39:15.0979 Model: WizardCaseBased
2013-05-10 10:39:15.0985 Model: WizardCaseBased
2013-05-10 10:39:15.0989 Model: WizardCaseBased
2013-05-10 10:39:15.0994 Model: WizardCaseBased
2013-05-10 10:39:15.0012 SubSample: 5
2013-05-10 10:39:15.0012 SubSample: 7
2013-05-10 10:39:15.0012 SubSample: 1
2013-05-10 10:39:15.0012 SubSample: 2
2013-05-10 10:39:15.0013 SubSample: 4
2013-05-10 10:39:15.0013 SubSample: 6
2013-05-10 10:39:15.0013 SubSample: 3
2013-05-10 10:39:16.0574 SubSample: 0
2013-05-10 10:39:16.0598 Reading Parameters
2013-05-10 10:39:16.0598 Reading Parameters
2013-05-10 10:39:16.0598 Reading Parameters
2013-05-10 10:39:16.0598 Reading Parameters
2013-05-10 10:39:16.0598 Reading Parameters
2013-05-10 10:39:16.0599 Reading Parameters
2013-05-10 10:39:16.0599 Reading Parameters
2013-05-10 10:39:16.0600 Reading Parameters
2013-05-10 10:39:16.0601 Running Simulation
2013-05-10 10:39:16.0602 Running Simulation
2013-05-10 10:39:16.0602 Running Simulation
2013-05-10 10:39:16.0602 Running Simulation
2013-05-10 10:39:16.0602 Running Simulation
2013-05-10 10:39:16.0602 Running Simulation
2013-05-10 10:39:16.0602 Running Simulation
2013-05-10 10:39:16.0603 Writing Output Tables
2013-05-10 10:39:16.0603 Running Simulation
2013-05-10 10:39:16.0605 Writing Output Tables
2013-05-10 10:39:16.0605 Writing Output Tables
2013-05-10 10:39:16.0606 Writing Output Tables
2013-05-10 10:39:16.0606 Writing Output Tables
2013-05-10 10:39:16.0607 Writing Output Tables
2013-05-10 10:39:16.0607 Writing Output Tables
2013-05-10 10:39:16.0607 Writing Output Tables
2013-05-10 10:39:16.0879 Done.
2013-05-10 10:39:16.0879 Done.
2013-05-10 10:39:16.0880 Done.
2013-05-10 10:39:16.0881 Done.
2013-05-10 10:39:16.0884 Done.
2013-05-10 10:39:16.0884 Done.
2013-05-10 10:39:16.0885 Done.
2013-05-10 10:39:16.0885 Done.

C:\om\prototype_lib\openm\Release\x64>sqlite3 -header -column WizardCaseBased.sqlite "SELECT Dim0, Value, Value0, Value
, Value2, Value3, Value4, Value5, Value6, Value7 FROM DurationOfLife ORDER BY 1"
Dim0        Value       Value0      Value1      Value2      Value3      Value4      Value5      Value6      Value7
----------  ----------  ----------  ----------  ----------  ----------  ----------  ----------  ----------  ----------
0           5000.0      5000.0      5000.0      5000.0      5000.0      5000.0      5000.0      5000.0      5000.0
1           0.0081      0.0081      0.0099      0.0013      0.0003      0.0061      0.0068      0.0192      0.0154
2           540.7658    540.7658    760.8044    688.7917    630.3369    750.2663    722.9372    645.1384    567.2142
3           71.9845     71.9845     73.2788     71.5934     71.5169     71.4986     69.9491     73.0923     70.8712
```




```
C:\om\prototype_lib\openm\Release\Win32>
C:\om\prototype_lib\openm\Release\Win32>
C:\om\prototype_lib\openm\Release\Win32>mpiexec -np 8 wizardCaseBased.exe
2013-05-10 10:36:26.0404 Model: WizardCaseBased
2013-05-10 10:36:26.0406 Model: WizardCaseBased
2013-05-10 10:36:26.0411 Model: WizardCaseBased
2013-05-10 10:36:26.0413 Model: WizardCaseBased
2013-05-10 10:36:26.0419 Model: WizardCaseBased
2013-05-10 10:36:26.0422 Model: WizardCaseBased
2013-05-10 10:36:26.0423 Model: WizardCaseBased
2013-05-10 10:36:26.0437 Model: WizardCaseBased
2013-05-10 10:36:26.0461 SubSample: 4
2013-05-10 10:36:26.0461 SubSample: 1
2013-05-10 10:36:26.0461 SubSample: 3
2013-05-10 10:36:26.0462 SubSample: 6
2013-05-10 10:36:26.0462 SubSample: 2
2013-05-10 10:36:26.0463 SubSample: 5
2013-05-10 10:36:26.0463 SubSample: 7
2013-05-10 10:36:26.0475 SubSample: 0
2013-05-10 10:36:26.0481 Reading Parameters
2013-05-10 10:36:26.0481 Reading Parameters
2013-05-10 10:36:26.0481 Reading Parameters
2013-05-10 10:36:26.0481 Reading Parameters
2013-05-10 10:36:26.0481 Reading Parameters
2013-05-10 10:36:26.0482 Reading Parameters
2013-05-10 10:36:26.0483 Reading Parameters
2013-05-10 10:36:26.0484 Reading Parameters
2013-05-10 10:36:26.0485 Running Simulation
2013-05-10 10:36:26.0485 Running Simulation
2013-05-10 10:36:26.0485 Running Simulation
2013-05-10 10:36:26.0486 Running Simulation
2013-05-10 10:36:26.0486 Running Simulation
2013-05-10 10:36:26.0486 Running Simulation
2013-05-10 10:36:26.0486 Running Simulation
2013-05-10 10:36:26.0486 Running Simulation
2013-05-10 10:36:26.0490 Writing Output Tables
2013-05-10 10:36:26.0490 Writing Output Tables
2013-05-10 10:36:26.0492 Writing Output Tables
2013-05-10 10:36:26.0493 Writing Output Tables
2013-05-10 10:36:26.0493 Writing Output Tables
2013-05-10 10:36:26.0494 Writing Output Tables
2013-05-10 10:36:26.0496 Writing Output Tables
2013-05-10 10:36:26.0497 Writing Output Tables
2013-05-10 10:36:27.0764 Done.
2013-05-10 10:36:27.0765 Done.
2013-05-10 10:36:27.0765 Done.
2013-05-10 10:36:27.0765 Done.
2013-05-10 10:36:27.0765 Done.
2013-05-10 10:36:27.0766 Done.
2013-05-10 10:36:27.0766 Done.
2013-05-10 10:36:27.0766 Done.

C:\om\prototype_lib\openm\Release\Win32>sqlite3 -header -column WizardCaseBased.sqlite "SELECT Dim0, Value, Value0, Valu
e1, Value2, Value3, Value4, Value5, Value6, Value7 FROM DurationOfLife ORDER BY 1"
Dim0        Value       Value0      Value1      Value2      Value3      Value4      Value5      Value6      Value7
----------  ----------  ----------  ----------  ----------  ----------  ----------  ----------  ----------  ----------
0           5000.0      5000.0      5000.0      5000.0      5000.0      5000.0      5000.0      5000.0      5000.0
1           0.0081      0.0081      0.0099      0.0013      0.0003      0.0061      0.0068      0.0192      0.0154
2           540.7658    540.7658    760.8044    688.7917    630.3369    750.2663    722.9372    645.1384    567.2142
3           71.9845     71.9845     73.2788     71.5934     71.5169     71.4986     69.9491     73.0923     70.8712
```

mpi@omm:~

```
login as: mpi
mpi@omm.beyond2020.com's password:
Last login: Fri May 10 11:13:35 2013 from wall.beyond2020.com
[mpi@omm ~]$
[mpi@omm ~]$ uname -a
Linux omm.beyond2020.com 2.6.32-358.6.1.el6.x86_64 #1 SMP Tue Apr 23 19:29:00 UTC 2013 x86_64 x86_64 x86_64 GNU/Linux
[mpi@omm ~]$
[mpi@omm ~]$ omrun -n 8 /mirror/wizardCaseBased
2013-05-10 11:18:39.0074 Model: WizardCaseBased
2013-05-10 11:18:39.0074 Model: WizardCaseBased
2013-05-10 11:18:39.0074 Model: WizardCaseBased
2013-05-10 11:18:39.0074 Model: WizardCaseBased
2013-05-10 11:18:39.0072 Model: WizardCaseBased
2013-05-10 11:18:39.0071 Model: WizardCaseBased
2013-05-10 11:18:39.0072 Model: WizardCaseBased
2013-05-10 11:18:39.0071 Model: WizardCaseBased
2013-05-10 11:18:40.0673 SubSample: 6
2013-05-10 11:18:40.0673 SubSample: 7
2013-05-10 11:18:40.0673 SubSample: 4
2013-05-10 11:18:40.0673 SubSample: 5
2013-05-10 11:18:40.0670 SubSample: 2
2013-05-10 11:18:40.0671 SubSample: 3
2013-05-10 11:18:40.0670 SubSample: 1
2013-05-10 11:18:41.0121 SubSample: 0
2013-05-10 11:18:41.0469 Reading Parameters
```

Beyond2020 HPC cluster

64bit

WizardCaseBased model, 8 sub-samples

...skip some output here...

```
2013-05-10 11:18:41.0894 Writing Output Tables
2013-05-10 11:18:45.0006 Done.
2013-05-10 11:18:45.0006 Done.
2013-05-10 11:18:45.0006 Done.
2013-05-10 11:18:45.0009 Done.
2013-05-10 11:18:45.0009 Done.
2013-05-10 11:18:45.0009 Done.
2013-05-10 11:18:45.0008 Done.
2013-05-10 11:18:45.0011 Done.
[mpi@omm ~]$
[mpi@omm ~]$ sqlite3 -header -column /mirror/WizardCaseBased.sqlite "SELECT Dim0, Value, Value0, Value1, Value2, Value
alue6, Value7 FROM DurationOfLife ORDER BY 1"
Dim0        Value       Value0      Value1      Value2      Value4      Value5      Value6      Value7
----------  ----------  ----------  ----------  ----------  ----------  ----------  ----------  ----------
0           5000.0      5000.0      5000.0      5000.0      5000.0      5000.0      5000.0      5000.0
1           0.0081      0.0081      0.0099      0.0013      0.0061      0.0068      0.0192      0.0154
2           540.7658    540.7658    760.8044    688.7917    750.2663    722.9372    645.1384    567.2142
3           71.9845     71.9845     73.2788     71.5934     71.4986     69.9491     73.0923     70.8712
[mpi@omm ~]$
```

Terminal - anatoly@anatoly... anatoly - File Manager

Terminal - anatoly@anatoly-xu1204-1: ~/prototype_lib/openm/bin/release

File Edit View Terminal Go Help

```
anatoly@anatoly-xu1204-1:~/prototype_lib/openm/bin/release$ uname -a
Linux anatoly-xu1204-1 3.2.0-41-generic #66-Ubuntu SMP Thu Apr 25 03:28:09 UTC 2013 i686 i686 i386 GNU/Linux
anatoly@anatoly-xu1204-1:~/prototype_lib/openm/bin/release$
anatoly@anatoly-xu1204-1:~/prototype_lib/openm/bin/release$ ./wizardCaseBased
2013-05-10 16:05:05.0187 Model: WizardCaseBased
2013-05-10 16:05:05.0874 SubSample: 0
2013-05-10 16:05:05.0878 Reading Parameters
anatoly@anatoly-xu1204-1:~/prototype_lib/openm/bin/release$ sqlite3 -column -header WizardCaseBased.sqlite "SELEC
FROM DurationOfLife ORDER BY 1"
Dim0        Value
----------  ----------
0           5000.0
1           0.0081
2           540.7658
3           71.9845
anatoly@anatoly-xu1204-1:~/prototype_lib/openm/bin/release$
```

Ubuntu Linux, non-clustered

32bit

WizardCasBased model, 1 sub-sample

```
gpc-f104n084-$ showq -w user=anatoly

active jobs------------------------
JOBID              USERNAME      STATE PROCS   REMAINING            STARTTIME
```

ComputeCanada SciNet GPC cluster: 32,508 CPU's

```
0 active jobs          0 of 32508 processors in use by local jobs (0.00%)
                     3863 of 3963 nodes active      (97.48%)

eligible jobs----------------------
JOBID              USERNAME      STATE PROCS     WCLIMIT            QUEUETIME

17574880            anatoly       Idle   512    00:15:00  Fri May 10 15:53:35

1 eligible job
```

...skip some output here...

```
2013-05-10 20:59:24.0657 Done.
2013-05-10 20:59:24.0938 Done.
----------------------------------------
Begin PBS Epilogue Fri May 10 20:59:32 EDT 2013 1368233972
Job ID:         17574880.gpc-sched
Username:       anatoly
Group:          mwolfson
Job Name:       wcb_test
Session:        22405
Limits:         neednodes=64:ppn=8,nodes=64:ppn=8,walltime=00:15:00
Resources:      cput=24:53:50,mem=24361588kb,vmem=99168844kb,walltime=00:04:19
Queue:          batch_ib
Account:
Nodes:  gpc-f103n024-ib0 gpc-f107n030-ib0 gpc-f107n036-ib0 gpc-f107n059-ib0
gpc-f112n082-ib0 gpc-f115n060-ib0 gpc-f118n070-ib0 gpc-f119n029-ib0
```

512 CPU's used by WizardCaseBased model

...skip some output here...

```
End PBS Epilogue Fri May 10 20:59:33 EDT 2013 1368233973
----------------------------------------
gpc-f104n084-$
gpc-f104n084-$ sqlite3 -column -header WizardCaseBased.sqlite 'SELECT Dim0, Value FROM DurationOfLife ORDER BY 1'
Dim0        Value
----------  ------------
0           1073741824.0
1           0.0
2           1485.3154
3           71.4271
gpc-f104n084-$
gpc-f104n084-$ sqlite3 -column -header WizardCaseBased.sqlite 'SELECT DISTINCT M.sub_id AS "SubSample",  (SELECT
tionOfLife_sub D0 WHERE D0.Dim0 = 0 AND D0.sub_id = M.sub_id) AS "Expr0",  (SELECT D1.sub_value FROM wizardcaseba
0 = 1 AND D1.sub_id = M.sub_id) AS "Expr1",  (SELECT D2.sub_value FROM wizardcasebased_1_DurationOfLife_sub D2 WH
) AS "Expr2",  (SELECT D3.sub_value FROM wizardcasebased_1_DurationOfLife_sub D3 WHERE D3.Dim0 = 3 AND D3.sub_id
sed_1_DurationOfLife_sub M ORDER BY 1' |more
SubSample   Expr0         Expr1       Expr2       Expr3
----------  ------------  ----------  ----------  ----------
0           1073741824.0  0.0         1485.3154   71.4271
1           1073741824.0  0.0         1485.3154   71.4268
2           1073741824.0  0.0         1534.8259   71.4294
3           1073741824.0  0.0         1534.8259   71.4279
4           1073741824.0  0.0         1534.8259   71.4288
```

WizardCaseBased model: 1,073,741,824 cases

WizardCaseBased: 512 sub-samples

...skip some output here...

```
509         1073741824.0  0.0         1534.8259   71.428
510         1073741824.0  0.0         1485.3154   71.4283
511         1073741824.0  0.0         1485.3154   71.4312
gpc-f104n084-$
```

2013, September: Alpha version

OpenM++ Alpha version: September 2013

OpenM++ alpha version released on September 2013. It includes:

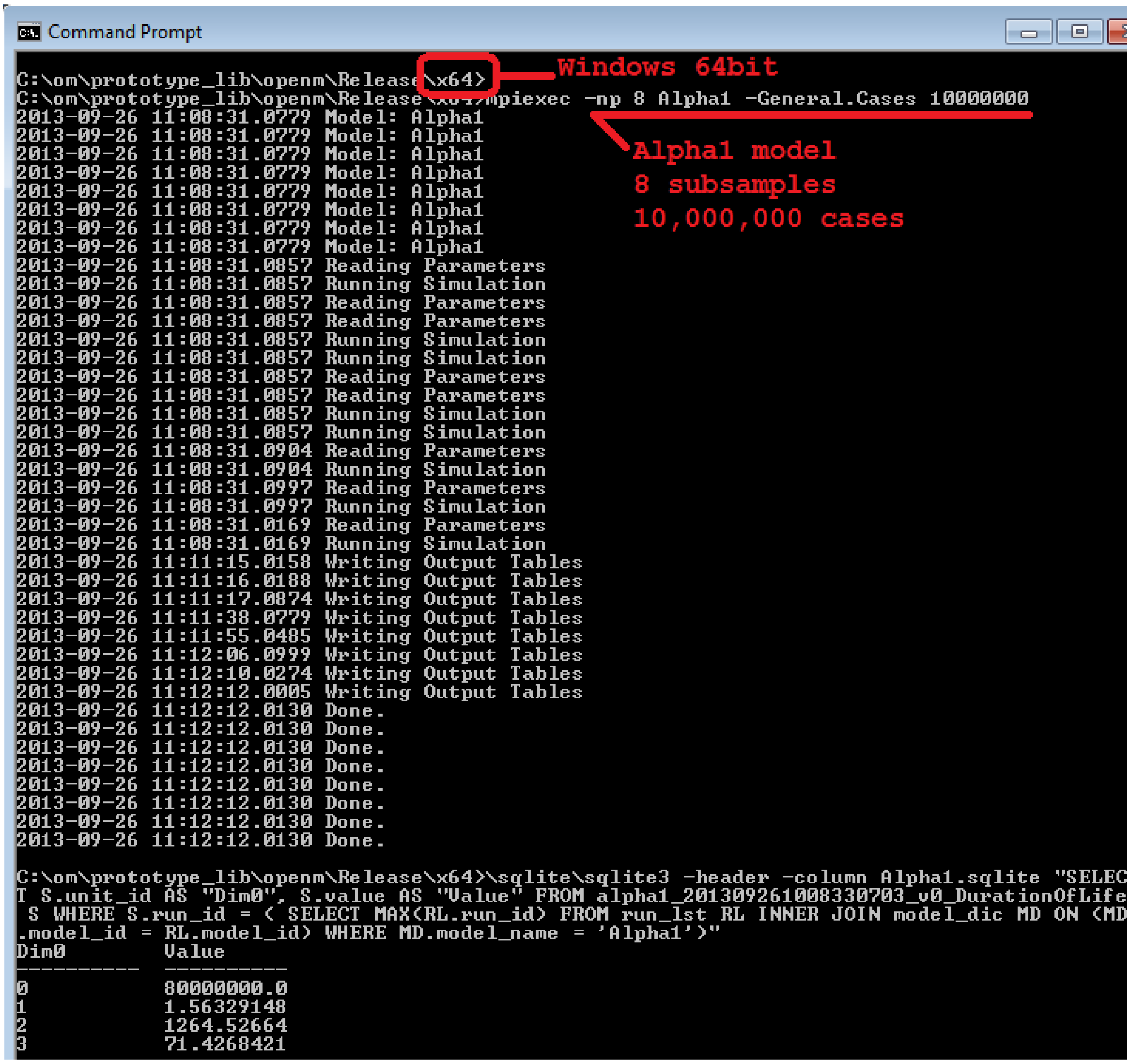
- openM++ compiler (alpha version)
- runtime library (combined model, data and execute libs)
- three models, compiled, build and running on all target platforms

Important results are:

- openM++ models are highly portable, zero efforts required to run same model on different platforms
- openM++ model produce identical results for all platforms and matching existing Modgen results

Below screenshots captured from openM++ Alpha1 model running on:

- Windows 64bit, 8 subsamples
- Windows 32bit, 8 subsamples
- Linux 64bit MPI cluster, 8 subsamples
- Linux 32bit non-clustered and without MPI, 1 subsample
- Linux 64bit HPC cluster at ComuputeCanada, 512 subsamples



Command Prompt

Windows 32bit

Alpha1 model
8 subsamples
10,000,000 cases

```
C:\om\prototype_lib\openm\Release\Win32>
C:\om\prototype_lib\openm\Release\Win32>mpiexec -np 8 Alpha1 -General.Cases 10000000
2013-09-26 10:56:55.0915 Model: Alpha1
2013-09-26 10:56:55.0931 Model: Alpha1
2013-09-26 10:56:55.0931 Model: Alpha1
2013-09-26 10:56:55.0931 Model: Alpha1
2013-09-26 10:56:55.0931 Model: Alpha1
2013-09-26 10:56:55.0931 Model: Alpha1
2013-09-26 10:56:55.0947 Model: Alpha1
2013-09-26 10:56:55.0947 Model: Alpha1
2013-09-26 10:56:55.0978 Reading Parameters
2013-09-26 10:56:55.0978 Running Simulation
2013-09-26 10:56:55.0978 Reading Parameters
2013-09-26 10:56:55.0978 Reading Parameters
2013-09-26 10:56:55.0978 Reading Parameters
2013-09-26 10:56:55.0978 Reading Parameters
2013-09-26 10:56:55.0978 Running Simulation
2013-09-26 10:56:55.0978 Running Simulation
2013-09-26 10:56:55.0978 Running Simulation
2013-09-26 10:56:55.0978 Running Simulation
2013-09-26 10:56:55.0056 Reading Parameters
2013-09-26 10:56:55.0056 Running Simulation
2013-09-26 10:56:55.0103 Reading Parameters
2013-09-26 10:56:55.0103 Running Simulation
2013-09-26 10:56:55.0212 Reading Parameters
2013-09-26 10:56:55.0212 Running Simulation
2013-09-26 11:01:03.0882 Writing Output Tables
2013-09-26 11:01:03.0412 Writing Output Tables
2013-09-26 11:01:04.0786 Writing Output Tables
2013-09-26 11:01:04.0958 Writing Output Tables
2013-09-26 11:01:05.0566 Writing Output Tables
2013-09-26 11:01:06.0565 Writing Output Tables
2013-09-26 11:01:06.0643 Writing Output Tables
2013-09-26 11:01:06.0204 Writing Output Tables
2013-09-26 11:01:06.0329 Done.
2013-09-26 11:01:06.0329 Done.
2013-09-26 11:01:06.0329 Done.
2013-09-26 11:01:06.0329 Done.
2013-09-26 11:01:06.0329 Done.
2013-09-26 11:01:06.0329 Done.
2013-09-26 11:01:06.0329 Done.
2013-09-26 11:01:06.0345 Done.

C:\om\prototype_lib\openm\Release\Win32>sqlite\sqlite3 -header -column Alpha1.sqlite "SEL
ECT S.unit_id AS "Dim0", S.value AS "Value" FROM alpha1_20130926100833O703_v0_DurationOfLi
fe S WHERE S.run_id = ( SELECT MAX(RL.run_id) FROM run_lst RL INNER JOIN model_dic MD ON (
MD.model_id = RL.model_id) WHERE MD.model_name = 'Alpha1')"
Dim0        Value
----------  ----------
0           80000000.0
1           1.56329148
2           1264.52664
3           71.4268421
```

mpi@omm:~

Beyond2020 Linux HPC cluster

64bit

Alpha1 model
8 subsamples
10,000,000 cases

```
login as: mpi
mpi@omm.beyond2020.com's password:
Last login: Thu Sep 26 13:40:37 2013 from wall.beyond2020.com
[mpi@omm ~]$
[mpi@omm ~]$ uname -a
Linux omm.beyond2020.com 2.6.32-358.6.2.el6.x86_64 #1 SMP Thu May 16 20:59:36 UTC 2013 x86_64 x86_64 x86_64 GNU/Linux
[mpi@omm ~]$
[mpi@omm ~]$ omrun -n 8 /mirror/alpha1 -General.Cases 10000000
2013-09-26 14:02:50.0627 Model: Alpha1
2013-09-26 14:02:50.0623 Model: Alpha1
2013-09-26 14:02:50.0624 Model: Alpha1
2013-09-26 14:02:50.0629 Model: Alpha1
2013-09-26 14:02:50.0630 Model: Alpha1
2013-09-26 14:02:50.0632 Model: Alpha1
2013-09-26 14:02:50.0629 Model: Alpha1
2013-09-26 14:02:50.0627 Model: Alpha1
2013-09-26 14:02:52.0332 Reading Parameters
```

...skip some output here...

```
2013-09-26 14:04:15.0698 Writing Output Tables
2013-09-26 14:04:18.0856 Done.
2013-09-26 14:04:18.0856 Done.
2013-09-26 14:04:18.0856 Done.
2013-09-26 14:04:18.0852 Done.
2013-09-26 14:04:18.0852 Done.
2013-09-26 14:04:18.0852 Done.
2013-09-26 14:04:18.0852 Done.
2013-09-26 14:04:18.0859 Done.
[mpi@omm ~]$
[mpi@omm ~]$ sqlite3 -header -column /mirror/Alpha1.sqlite "SELECT S.unit_id AS "Dim0", S.value AS "Value" FROM alpha1
_20130926132811O855_v0_DurationOfLife S WHERE S.run_id = ( SELECT MAX(RL.run_id) FROM run_lst RL INNER JOIN model_dic
MD ON (MD.model_id = RL.model_id) WHERE MD.model_name = 'Alpha1')"
Dim0        Value
----------  ----------
0           80000000.0
1           1.56329148
2           1264.52664
3           71.4268421
[mpi@omm ~]$
```

```
anatoly@anatoly-xu1304-1: ~/prototype_lib/openm/bin/release
anatoly@anatoly-xu1304-1:~/prototype_lib/openm/bin/release$ uname -a
Linux anatoly-xu1304-1 3.8.0-31-generic #46-Ubuntu SMP Tue Sep 10 19:56:49 UTC 2013 i686 i686 i686 GNU/Linux
anatoly@anatoly-xu1304-1:~/prototype_lib/openm/bin/release$
anatoly@anatoly-xu1304-1:~/prototype_lib/openm/bin/release$
anatoly@anatoly-xu1304-1:~/prototype_lib/openm/bin/release$
anatoly@anatoly-xu1304-1:~/prototype_lib/openm/bin/release$
anatoly@anatoly-xu1304-1:~/prototype_lib/openm/bin/release$ ./alpha1 -General.Cases 10000000
2013-09-27 10:55:40.0071 Model: Alpha1
2013-09-27 10:55:40.0102 Reading Parameters
2013-09-27 10:55:40.0102 Running Simulation
2013-09-27 10:56:26.0308 Writing Output Tables
2013-09-27 10:56:26.0334 Done.
anatoly@anatoly-xu1304-1:~/prototype_lib/openm/bin/release$ sqlite3 -header -column Alpha1.sqlite "SELECT S.unit_id AS "Dim0", S.value AS "Value" FROM alpha1_201309271035060599_v0_DurationOfLife S WHERE S.run_id = ( SELECT MAX(RL.run_id) FROM run_lst RL INNER JOIN model_dic MD ON (MD.model_id = RL.model_id) WHERE MD.model_name = 'Alpha1')"
Dim0        Value
----------  ----------
0           10000000.0
1           8.48168823
2           1264.52664
3           71.4589877
anatoly@anatoly-xu1304-1:~/prototype_lib/openm/bin/release$
```

Ubuntu Linux, non-clustered, 32bit

Alpha1 model,
1 subsample,
10,000,000 cases

```
gpc-f102n084-$
gpc-f102n084-$ showq -w user=anatoly

active jobs------------------------
JOBID              USERNAME      STATE  PROCS   REMAINING            STARTTIME


0 active jobs            0 of 32580 processors in use by local jobs (0.00%)
                       3881 of 3972 nodes active       (97.71%)

eligible jobs----------------------
JOBID              USERNAME      STATE  PROCS     WCLIMIT            QUEUETIME

20703912            anatoly       Idle    512     1:00:00  Mon Nov 18 08:42:57

1 eligible job
```

ComputeCanada SciNet GPC cluster: 32,580 CPU's

```
2013-11-18 10:11:50.0742 Done.
2013-11-18 10:11:50.0744 Done.
----------------------------------------
Begin PBS Epilogue Mon Nov 18 10:11:56 EST 2013 1384787516
Job ID:         20703912.gpc-sched
Username:       anatoly
Group:          mwolfson
Job Name:       alpha1_test
Session:        25636
Limits:         neednodes=64:ppn=8,nodes=64:ppn=8,walltime=01:00:00
Resources:      cput=11:12:20,mem=8500080808kb,vmem=143316160kb,walltime=00:03:10
Queue:          batch_ib
Account:
Nodes:  gpc-f109n030-ib0 gpc-f109n031-ib0 gpc-f109n032-ib0 gpc-f109n033-ib0
gpc-f109n034-ib0 gpc-f109n035-ib0 gpc-f109n037-ib0 gpc-f109n038-ib0
```

512 CPU's used by Alpha1 model

```
End PBS Epilogue Mon Nov 18 10:11:57 EST 2013 1384787517
----------------------------------------
gpc-f102n084-$
gpc-f102n084-$ sqlite3 -column -header Alpha1.sqlite "SELECT S.unit_id AS "Dim0", S.value AS "Value" FROM alpha1_201311161801100756_v0_DurationOfLife S WHERE S.run_id = (SELECT MAX(RL.run_id) FROM run_lst RL INNER JOIN model_dic MD ON (MD.model_id = RL.model_id) WHERE MD.model_name = 'Alpha1')"
Dim0        Value
----------  ------------
0           5120000000.0
1           3.3261520529
2           1534.8258997
3           71.428337802
gpc-f102n084-$
```

Alpha1 model
512 subsamples
10,000,000 cases

2014, March: Project Status, Phase 1 completed

Current Project Status

OpenM++ phase 1 completed in March 2014 with following results (enumeration corresponds to OpenM++ design document, which also describes tasks):

- OpenM++ compiler (2.1 priority1): alpha version, working beta version with 60% functionality coverage
  - types (classifications, ranges, partitions)
  - parameters (exogeneous)
  - agents
  - variables (25% complete)
  - inter-agent links
  - events
  - cross tabulation (except margins)
  - meta-information (except labels & groups)
- OpenM++ controller for MPI cluster (2.2 priority1): beta version
- OpenM++ modelling library (3.1 priority1): beta version
  - case-based and time-based models
  - agent & event lifecycle
  - event queue
  - on-the-fly cross-tabulation updating
  - Modgen-equivalent random number generators for exact output comparability
- OpenM++ model data storage design (3.2 priority1): beta version
- OpenM++ data library (3.3. priority1): beta version
- OpenM++ execute library (3.4 priority1): beta version

On top of essential phase 1 Roadmap tasks following items completed:

- compatibility layer for Modgen source model code .mpp files (2.3 priority2): alpha version
- OpenM++ output result viewers and model analysis tools, import/export into R (1.2 priority2): beta version

Deferred items mentioned in phase 1 Roadmap:

- all optional items (except two listed above) due to limited resources
- compatibility converter for Modgen parameters .dat files (2.3 priority1) postponed after extensive design discussions.
- components of OpenM++ compiler (2.1 priority1) due to limited resources
  - agent collections
  - parameters (endogenous)
  - variables (75% remaining)
  - cross-tabulation (margins)
  - meta-information (labels & groups)
  - derived tables
  - other miscellaneous functionality

Overall results of OpenM++ phase 1 cover most of existing Modgen desktop functionality (excluding GUI).

What Next

OpenM++ foundation created as result of phase 1 project and it is opens up following four streams for subsequent development:

- model stream: ongoing work to move existing Modgen models onto OpenM++ platform
- cloud stream: build openM++ cloud PaaS and/or SaaS stack, emphasizing on scalability. **Time:** 11 months
- tools stream: creating openM++ desktop GUI based on Visual Studio, Eclipse or similar for model developers and users. **Time:** 9 months
- core stream: enhance openM++ core functionality, for example, modelling results post-processing and analysis.

Tools and cloud stream partially described in OpenM++ Design and Model Architecture documents.

Core stream task list is very flexible because it is generally include OpenM++ small core enhancements required for other development streams.

For example, as it is today, beta version of OpenM++ supports only SQLite as model database storage and cloud version of OpenM++ most likely require at least one of MySQL, PostgreSQL, Oracle, MSSQL or DB2 support. Due to flexible nature of core stream development it can be done incrementally as long as resources available, however it is very important strictly follow OpenM++ Design documents to make sure we are proceeding to the right direction and avoid common "creeping featurism" mistake.

Current List of small tasks

Following tasks are required to be completed before or during OpenM++ cloud or desktop GUI development (enumeration corresponds to OpenM++ design):

- OpenM++ output converters:
  - 2.5 priority 1: export into .csv for parameters and output results. **Time:** 10 days
  - 2.5 priority 2: export into .xml for model data or user-defined subset of model data. **Time:** 16 days
- OpenM++ SQL loaders. **Time:** 4 weeks + 10 days for each db-vendor
  - 2.4 priority 1: MS SQL, MySQL / MariaDB
  - 2.4 priority 2: generic SQL99
  - 2.4 priority 3: PostgreSQL, Oracle, DB2, Apache Derby, H2 or HSQL
- extend data library to support MySQL, PostgreSQL, Oracle, MSSQL or DB2 (3.3 priority3). **Time:** 3-4 weeks for each db-vendor
- completion of OpenM++ core support for i18n / L10n in runtime library. **Time:** 3 weeks
- Modgen .dat files compatibility converter (2.3 priority 1): required design decision. **Time:** from 10 days to 6 weeks.
- exploratory subsamples suite for OpenM++ models (see below). **Time:** between 5-9 weeks

Their is no fixed order in the list above, it can be implemented as required by other project or OpenM++ users.

Task: Modgen .dat files compatibility converter

This is high priority component which defined in OpenM++ design document (2.3 priority 1) as command-line utility to convert existing Modgen models data into OpenM++ format. It was originally planned for phase 1 development, but deferred due to unresolved design dependency with other parts of OpenM++, i.e. cloud model publusher or SQL loaders mentioned above.

There are two important aspects of .dat-convertor design:

- language complexity of .dat file syntax, which is in fact c++ initializers syntax with Modgen extensions
- environmental complexity of .dat-convertor use cases

Environmental complexity actually means variety of .dat-convertor use case scenarios in not yet well defined runtime environment. Please look at explanation on OpenM++ model use cases in Model Architecture document for more details.

Some examples may include:

- developer:

    - uses local Windows or Linux PC with GUI
    - often recreate SQLite database and load input data hundred times to debug the model
    - eventually need to pack model executable and data files and send it to researcher
- researcher:
    - HPC cluster (large or small) or local Windows, Linux workstation without GUI
    - run the model thousand times loading wide variety of input data from prepared .dat files
    - do not have admin privileges, especially on cluster, as result, can not install or adjust runtime environmemnt
    - often need to pack model .dat files to publish it, move from local PC to HPC cluster or share with other researchers
- institutional user:
    - uses web UI to run the model in cloud, on HPC cluster or other powerful server environment
    - have absolutely no access to actual server environment
    - receives initial set of input .dat files from developer or researcher and want to upload it into cloud database
    - cloud database most likely one of: MySQL, Oracle, MSSQL, PostgreSQL, DB2

From examples above you can see following requirements to model input data tools:

- it must be portable and can not assume specific OS or database
- user may have no access to actual model database (i.e. model developer have no access to cloud instance)

Possible solutions for .dat-files converter in context of above requirements:

- due to language complexity of .dat files it is nice to use OpenM++ compiler (omc) to parse it
- omc read .dat files and saves as:
    - c++ values compiled into model executable, which in turn, saves it into target database during first run
        - pro: everything in one file, ideal for consistency and transfer
        - cons: model executable is huge, which increase compilation and execution time
        - pro/cons: it is not possible to change model input data without re-compilation
    - SQLite database
        - pro: compact storage
        - pro: ideal for model developer (or even researcher) as no any other steps required to run the model
        - pro: there are many standard utilities to browse or edit the data
        - cons: extra tool required to import from SQLite into actual database in cloud environment
    - sql script files
        - pro: portable and human-readable format
        - pro: no any other tools required to transfer data from one environment into another
        - cons: least compact storage, size of input data files are largest of all
    - some other text format, i.e.: .csv or .xml files
        - pro: portable and human-readable format
        - cons: some custom tools required to load the data from such files into model database

We must keep in mind when choosing .dat-converter solution couple of other items from OpenM++ design document:

- OpenM++ must have SQL-loader utilities to facilitate data export into different model databases
- OpenM++ must have utilities to export input (and output) data into other formats, i.e.: text .csv and .xml files

That means we can relay on presence such OpenM++ utilities in foreseeable future.

Task: Exploratory subsamples suite for OpenM++ models

Current OpenM++ model subsamples design is Modegn-compatible. It was done on purpose to provide Modgen model developers and users familiar concepts and even ability to reuse existing tools within OpenM++. However, there is fundamental limitation in that design, which became obvious when OpenM++ model runs in HPC cluster environment.

For example, if we have 16,000 CPUs cluster then it may be make sense to prepare 1000 different sets of input parameters, submit model job with those 1000 inputs * 16 subsamples each to use all 16,000 CPUs and analyse results to find optimal set of model parameters. It is possible to do in OpenM++ now by specifying working set of input parameters for and run the model 1000 times by submitting 1000 jobs to the cluster. However it would be nice to have such capability incorporated in OpenM++ runtime to allow simply submit single job with 1000 different sets of parameters and 16 subsamples each.

To implement such feature following changes in OpenM++ required:

- execution library: organize model MPI groups to effectively broadcast input parameters to slave modelling processes
- model database schema: allow multiple sets of input parameters for each model run (Modgen allow only single)
- model database schema: store relationship between input set and output results inside of single modelling job
- data library: redesign output results aggregation to do it over related output values (now it is done across all subsamples)

That feature should significantly increase model users productivity and allow more effective HPC cluster resource usage. It is recommended to have it for OpenM++ cloud version.

2016, December: Task List

Their is no fixed order in the list below, it can be implemented as required by other project or OpenM++ users.

Soft simulation failure

Currently any model exception is a hard failure and result in model shutdown. It may be right things to do in most situations but can be soften for some simulation errors. If special `SimulationException` thrown by model it should only abort current model run (or even current subsample only) and allow to proceed with next run from modeling task (or next subsample).

Write fixed model parameters in database

Currently fixed (or model generated or derived) model parameters not saved in database and completely hidden from model user. It is a good feature of openM++ in terms of data security, but may not be always necessary. It would be nice to have special openM++ language clause which model developer can use to control when fixed (or model generated or derived) parameter completely hidden or written in database as "output read-only parameter".

Write only selected output tables in database

Currently all output tables written in database as result of model run, which may be unnecessary if user doing parameter estimation and interested only in one or small subset of output tables. It would be nice to have an ability to specify which output tables needs to be written in database.

2017, January: Design Notes. Subsample As Parameter problem. Completed

Status: completed

Task is completed, notes below we do keep just in case.

Problem Scope

This is design notes, it is sketchy and may be incorrect, feel free to change it.

Currently we have one special model parameter: subsample number (a.k.a. member or replica). It is created by runtime as integer [0,N] where N is number of subsamples specified as run option:

```
model.exe -General.Subsamples 16
```

Subsample number plays fundamental role in calculation of model Output Expressions. It is only parameter which used to calculate average (CV, SE, SD and all others) output values. For example if model runs with 16 subsamples then it will produce 16 values for each output accumulator and output expression value is an average of 16 accumulators across subsamples.

It may not be always necessary to have subsample number as special parameter; it can be any other model parameter or set of parameters which varies between model runs. And output expression(s) can be calculated as average (CV, SD, etc.) across any parameter values. However such "demote of subsample number" is quite significant change in model runtime.

Currently model run cycle looks like (extremely simplified):

- start model.exe and connect to database
- read all model parameters
- create modeling threads for each model subsample
- run modeling threads: do simulation
- write output accumulators for each subsample in database
- wait until all subsamples done (wait for exit from all modeling threads)
- calculate output expression values as average (CV,SE,SD,etc.) of accumulators across subsamples
- report on simulation success and exit from model main

If we decide to "demote subsample" or call it as "generalize parameters" then modeling cycle can look like:

- use some external utility to create modeling task and prepare set of input parameter (see Model Run: How to Run the Model)
- (optional) specify runtime expression to vary some model parameters, e.g. subsample number parameter
- run model until modeling task completed (until all input processed) and write all accumulators into database
- use some external utility to calculate output expressions as average (CV,SE,SD,etc.) across any parameter(s)

Questions and problems:

1. How to specify model parameters generators (how to calculate model parameters at runtime). Now we have ompp code translated into c++ by omc compiler to do all derived (model-generated) parameters. It is not dynamic enough - we don't want and should not re-compile model to specify parameter(s) generator. We also have primitive subsample number parameter generator as [0,N]. Such primitive for-loop generators may be good in many situations but not enough.

Is it enough to have an ability in model runtime specify for-loop parameter(s) generator(s) and rely on external utilities (i.e. use our R package) to create more complex modeling tasks?

2. Output expressions calculations. Now we use SQL to calculate averages and, in fact, that SQL allow to have almost arbitrary calculation, but it does aggregation across subsample number.

How to generalize SQL to aggregate across any parameter values, not only subsample number? Do we need to replace SQL with c++ code in model runtime? Do we need to create other "db_aggregator" utility instead of using model?

3. How to specify parameter generators and output expressions to make it powerful enough and avoid re-inventing of R (Octave, Matlab, SPSS, SAS)?

Example of the problem

Let's assume some hypothetical model with following input parameters:

- population by age and sex
- taxation level
- election outcome
- workforce strike longevity
- random generator seed And model output value is household income.

Model input parameters can be divided in following categories:

- "constant": where parameter values are known and does not changed during modeling
  - population current and projected values assumed to be well known and fixed for our model
- "variable": parameter(s) which user want to change to study effect on modeling output results
  - taxation level varies from 1% to 80% with 0.1% step
- "uncertainty": parameters where values are random
  - election outcome parameter: Bernoulli distribution (binary) with mean = 0.6
  - workforce strike: Poisson distribution with rate = 4
  - random number generator seed

In order to study taxation level effect user run the model 800 times with different tax percent input value and calculate 800 average household income output values. Each output income value is an average of 32 "accumulator" values. Each "accumulator" value is a household income value produced by the model for specific combination of "uncertainty" parameters:

```
// create 32 input tuples of uncertainty parameters
//
int setId = database.CreateWorkset(); // input set of uncertainty parameters
bool isBluePartyWin = false;          // election results: win of "blue" or "red" party
double strikeDays = 7.5;              // number of strike days per year
int randomSeed = 12345;               // random number generator seed

for (int k = 0; k < 32; k++) {
  isBluePartyWin = Bernoulli(0.6);
  strikeDays = SumOf_Poisson(4.0);
  seed++;
  // write "uncertainty" parameters into database input set: tuple number = k
  database.WriteParameters(setId, k, isBluePartyWin, strikeDays, randomSeed);
}

// run the model
//
for (double tax = 1; tax < 82; tax += 0.1) {
  model.exe -Parameter.Taxation tax -UncertaintyParameters setId
}
//
// plot output household income depending on taxation level
//
```

Pseudo code above can be implemented in Perl, R or using shell script. Also openM++ already support Modeling Task which allow to submit multiple inputs to the model and vary parameter(s) values similar to example above.

Solution overview

OpenM++ already have most of components required for our solution, please take a look at:

- Modeling Task
- Input parameters sets (workset)

- Results aggregation: Model Output Expressions

Following can be done to solve a problem from example above:

1. **Use existing:** R API to create Modeling Task with 800 values of taxation level parameter.

2. **Add new:** Create tools to generate uncertainty parameters. It can be command-line utilities, GUI tool(s) or part of model runtime. Last option would allow us to reuse existing c++ code.

3. **Add new:** Change database schema in order to store tuples of uncertainty parameters as part of model run input. Currently model is using only single input set of parameters (workset) with single value of each parameter. We need to change database schema and model run initialization (input parameters search in database) in order to supply all 32 tuples of uncertainty parameters for every model run.

4. **Add new:** Change parameters memory management in order to provide unique value of each uncertainty parameter to each modeling thread. Now all parameters have only one copy of values and it is shared between all subsamples (threads and processes); only subsample number is unique and not shared between threads (see model run on single computer). And with new runtime we need to make sure only "constant" and "variable" parameters (like population and taxation level above) are shared and "uncertainty" parameters (election outcome, strike, random seed) are unique for each thread.

5. **Add new:** In case if model run on MPI cluster, when there are multiple modeling processes, we need to correctly supply unique values of all uncertainty parameters to each process. Now only subsample number is unique.

6. **Add new:** Change database schema similar to (3) above for model run parameters. Model run contains full copy of input parameters. Today it is only one value for each parameter and we need to change it in order to store all 32 tuples of uncertainty parameters in model run results.

7. **Use existing:** Model Output Expressions for output results aggregation. No changes required. We not yet have capabilities to compare model run results similar to what ModgenWeb does, but this is out of problem scope.

We can split implementation into two steps:

- First do all necessary run time changes (items 3, 4, 5 and 6 above). That would allow us to run the model with uncertainty parameters created by external tools, for example by R.

- Second is to implement "parameters generators" (item 2 above) to make it convenient to model user.

During that two steps process it is also necessary to implement some compatibility logic to supply parameter "Subsample" in order to keep existing models working.

Note: We should also solve ambiguity of "subsample" term, inherited from Modgen. It can be a model integer parameter with name "Subsample" and in that case it same as any other model parameter, no any kind of special meaning or treatment required. It is also can be used as "uncertainty tuple number" and may not be necessary exposed to modeling code, it can be internal to model runtime and visible in database schema as `sub_id` to order accumulator values and make it comparable between model runs.

Oms: openM++ web-service

What is openM++ web-service

OpenM++ web-service (oms) is a JSON web-service written in Go and used from openM++ UI JavaScript. Today most of popular development platforms (.NET, Java, Python, Perl, R, JavaScript, etc.) with only few lines of code allow to create HTTP client and send-receive JSON data. That makes integration with openM++ very easy.

How to start openM++ web-service

OpenM++ web-service does not required any installation. It can be run with default settings from command-line prompt.

To start openM++ web-service on Windows:

- download and unzip openM++ https://github.com/openmpp/main/releases/latest binaries into `C:\SomeDir\`
- run oms from command-line:

```
C:
cd \SomeDir\openmpp_win_20190508\
bin\oms.exe
```

```
2022-09-14 15:51:30.477 Models directory:    models\bin
2022-09-14 15:51:30.565 HTML UI directory:   html
2022-09-14 15:51:30.567 Etc directory:       etc
2022-09-14 15:51:30.567 Oms instance name:   localhost_4040
2022-09-14 15:51:30.574 Listen at localhost:4040
2022-09-14 15:51:30.574 To start open in your browser: http://localhost:4040
2022-09-14 15:51:30.574 To finish press Ctrl+C
```

OpenM++ UI is a client of `oms` web-service, after above command you can open UI in browser at http://localhost:4040

To start openM++ web-service on Linux:

- download and unpack openM++, i.e.:

```
wget https://github.com/openmpp/main/releases/download/v1.2.0/openmpp_debian_20190508.tar.gz
tar xzf openmpp_debian_20190508.tar.gz
```

- run oms executable:

```
cd openmpp_debian_20190508/
bin/oms
```

```
2022-09-14 15:51:30.477 Models directory:    models/bin
2022-09-14 15:51:30.565 HTML UI directory:   html
2022-09-14 15:51:30.567 Etc directory:       etc
2022-09-14 15:51:30.567 Oms instance name:   localhost_4040
2022-09-14 15:51:30.574 Listen at localhost:4040
2022-09-14 15:51:30.574 To start open in your browser: http://localhost:4040
2022-09-14 15:51:30.574 To finish press Ctrl+C
```

Note: We recommend to use normal Windows command line cmd.exe. If you are using Windows PowerShell then it may be necessary to put "quotes" around command line options, e.g:

```
oms.exe "-oms.ApiOnly"
```

Oms as "pure" web-service vs "full" web-UI

By default `oms.exe` started in "full" web-UI mode. That means it handles web-service requests and web-UI content from `./html` sub-directory. If you want only "pure" web-service mode without UI then use:

```
oms -oms.ApiOnly
```

How to use oms: arguments of web-service methods

Following arguments most often used in web-service methods:

:model - model digest or model name

Example of method:

```
GET /api/model/:model
```

Call example:

```
http://localhost:4040/api/model/f5024ac32c4e8abfc696a0f925141c95
http://localhost:4040/api/model/modelOne
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

:run - model run or model task run

Example of method:

```
GET /api/model/:model/run/:run/status
GET /api/model/:model/task/:task/run-status/run/:run
```

Call example:

```
http://localhost:4040/api/model/modelOne/run/modelOne_first_run/status
http://localhost:4040/api/model/modelOne/run/d06f4a0a45a9514c22593025e489f933/status
http://localhost:4040/api/model/modelOne/task/taskOne/run-status/run/First Task Run
```

This argument is used to identify model run or modeling task run.

Modeling task run can be identified by task run stamp or task run name.

Model run can be identified by run digest, run stamp or run name. It is recommended to use run digest because it is uniquely identifies model run. Run stamp can be explicitly specified as command line option when you run the model. If run stamp not specified then it is automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use run name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

:lang - language code

Example of method:

```
GET /api/model/:model/text/lang/:lang
```

Call example:

```
http://localhost:4040/api/model/modelOne/text/lang/EN
http://localhost:4040/api/model/modelOne/text/lang/en_US
```

Language code can be a model language (ex.: EN, FR) or any MIME language (see BCP47 or RFC3282). If no language explicitly specified then `Accept-Language` header is used (supplied by browser).

Result returned in best matched language supported by model. For example for en_US result is model language EN, if model supported EN language. If no such language then result is in default model language or can be empty.

:set - set of input data (a.k.a. workset)

Method examples:

```
GET  /api/model/:model/workset/:set/status
POST /api/model/:model/workset/:set/readonly/:val
```

Call examples:

```
http://localhost:4040/api/model/modelOne/workset/modelOne_set/status
curl -v -X POST http://localhost:4040/api/model/modelOne/workset/modelOne_set/readonly/1
```

Workset is a set of model input parameters (a.k.a. "scenario" input) and it used to run the model Each model workset uniquely identified by name.

:task - modelling task

Method examples:

```
GET /api/model/:model/task/:task/text/lang=FR
```

Call examples:

```
http://localhost:4040/api/model/modelOne/task/taskOne/text
curl -v http://localhost:4040/api/model/modelOne/task/taskOne/text/lang=fr_CA
```

Modelling task consists of multiple input data sets (a.k.a. worksets or scenarios in Modgen). Task can be used to run the model in batch mode.

:profile - set of key-value options

Method examples:

```
GET /api/model/:model/profile/:profile
POST /api/model/:model/profile/:profile/key/:key/value/:value
```

Call examples:

```
http://localhost:4040/api/model/modelOne/profile/modelOne
curl -v -X POST http://localhost:4040/api/model/modelOne/profile/m1/key/Parameter.StartingSeed/value/4095
```

Profile is a set of key-value options and it used to run the model. Each profile uniquely identified by profile name. Each profile can include multiple key-value options.

Results of web-service methods

Run status

Model run status and task run status may contain one of the following values:

```
 i = initial state, not running yet
 p = run in progress
 w = task wait for additional input
 s = completed successfully
 x = completed by exit (reserved fro internal use)
 e = completed with error
```

Important: if model run failed with exception (e.g. database write exception) then status may not be updated and still `p=in progress`.

Oms web-service configuration

Oms default configuration options can be overwritten by command-line arguments or ini-file. For example:

- listen from any host on port 7070:

  ```
  oms -l :7070
  ```

- serve only API calls and not `html` for openM++ UI:

  ```
  oms -oms.ApiOnly
  ```

- listen from localhost port 4044 only and read more oms run options from `oms.ini` file:

  ```
  oms -l localhost:4044 -ini oms.ini
  ```

- models directory relative path is: `../some/dir`

  ```
  oms -oms.ModelDir ../some/dir
  ```

- typical log settings for remote server:
  - log user request
  - log into the file instead of cosole by default
  - log files rotation: create new log file every day

    ```
    oms -l localhost:4044 -oms.LogRequest -OpenM.LogToConsole false -OpenM.LogToFile -OpenM.LogUseDailyStamp
    ```

- typical settings for model user in cloud:
  - allow user home directory with downloads and uploads
  - use model run jobs to manage back-end computational servers resources

    ```
    oms -l localhost:4044 -oms.HomeDir models/home -oms.AllowDownload -oms.AllowUpload -oms.JobDir job
    ```

    It is recommended to use `oms.ini` file to avoid long command lines, especially for cloud environment where you may want to combine log options and user options from two examples above.

Get and use oms web-service configuration

Clients of oms web-service can retrieve configuration by calling GET web-service configuration or simply by open http://localhost:4040/api/service/config in the browser. Response to that call may also contain client environment variables which names started from `OM_CFG_` prefix *(`oms` web-service does not use any of `OM_CFG_` environment variables, it only passes it to clients)*.

For example openM++ UI uses following server variables:

```
OM_CFG_LOGIN_URL=/public/login_required.html
OM_CFG_LOGOUT_URL=/login?logout=true
OM_CFG_DEFAULT_RUN_TMPL=run.Win32.Debug.template.txt
OM_CFG_INI_ALLOW=true
OM_CFG_INI_ANY_KEY=true
```

OpenM++ UI is using above variables as follow:

- `OM_CFG_LOGIN_URL` : display user login button linked to the URL
- `OM_CFG_LOGOUT_URL` : display user logout button linked to the URL
- `OM_CFG_DEFAULT_RUN_TMPL` : use this template to run the model, e.g.: to debug from IDE
- `OM_CFG_INI_ALLOW` : allow user to run the model with ini-file, e.g.: `RiskPaths.ini`
- `OM_CFG_INI_ANY_KEY` : allow to use model development options from ini-file

Note: Model ini-files and model development options described at: Model Run Options and ini file.

Oms run options

Following options supported by oms:

```
-oms.Listen:        address to listen, default: localhost:4040
-l:                 address to listen (short form of -oms.Listen)
-OpenM.IniFile:     path to ini-file
-ini ini-file:      path to ini-file (short of OpenM.IniFile)
-oms.ApiOnly:       if true then API only web-service, no web UI
-oms.RootDir:       oms root directory, default: current directory
-oms.ModelDir:      models directory, if relative then must be relative to root directory, default: models/bin
-oms.ModelLogDir:   models log directory, if relative then must be relative to root directory: default: "models/log
-oms.HomeDir:       user personal home directory, if relative then must be relative to root directory
-oms.AllowDownload: if true then allow download from user home/io/download directory
-oms.AllowUpload:   if true then allow upload to user home/io/upload directory
-oms.HtmlDi:        front-end UI directory, if relative then must be relative to root directory, default: html
-oms.EtcDir:        configuration files directory, if relative then must be relative to oms root directory, default: etc
-oms.JobDir:        model run jobs directory, if relative then must be relative to root directory
-oms.Name:          oms instance name, used model run by jobs, automatically generated if empty
-oms.UrlSaveTo:     file path to save oms URL in form of: http://localhost:4040, if relative then must be relative to root directory
-oms.Languages:     comma-separated list of supported languages, default: en
-oms.CodePage:      code page to convert source file into utf-8, e.g.: windows-1252
-oms.DoubleFormat:  format to convert float or double value to string, default: %.15g
-oms.MaxRowCount:   max number of rows to return from read parameters or output tables, default: 100
-oms.MaxRunHistory: max number of model runs to keep in run list history, default: 1000

-OpenM.LogToConsole:     if true then log to standard output (default true)
-v:                      if true then log to standard output (short of OpenM.LogToConsole)
-OpenM.LogToFile:        if true then log to file
-OpenM.LogFilePath:      path to log file, default = current/dir/oms.log
-OpenM.LogUseDailyStamp: if true then use daily-stamp in log file name
-OpenM.LogUsePidStamp:   if true then use process id stamp in log file name
-OpenM.LogUseTimeStamp:  if true then use time-stamp in log file name
-OpenM.LogSql:           if true then log sql statements into log file
-oms.LogRequest:         if true then log HTTP requests
```

There are many common options, e.g.: `-OpenM.LogToFile` which can be used with any openM++ executable: models, compiler, dbcopy and oms.

Example of oms.ini

```
; This is a comment, started with ; semicolon
# This is a comment, started with # hash
; boolean options can be "true" or "false"

[oms]
Listen        = localhost:4040 # address to listen, default: localhost:4040
RootDir       =                # oms "root" directory, expected to have log subfolder
ModelDir      = models/bin     # models executable and model.sqlite directory, if relative then must be relative to oms root directory
ModelLogDir   = models/log     # models log directory, if relative then must be relative to oms root directory
HomeDir       = models/home    # user personal home directory, if relative then must be relative to oms root directory
AllowDownload = false          # if true then allow download from user home sub-directory: home/io/download
AllowUpload   = false          # if true then allow upload to user home sub-directory: home/io/upload
ApiOnly       = false          # if true then API only web-service, no web UI
HtmlDir       = html           # front-end web UI directory, if relative then must be relative to oms root directory
EtcDir        = etc            # configuration files directory, if relative then must be relative to oms root directory
JobDir        =                # jobs control directory, if empty then jobs control disabled
Name          =                # instance name, used for job control
UrlSaveTo     =                # file path to save oms URL, if relative then must be relative to oms root directory
LogRequest    = false          # if true then log HTTP requests
Languages     = en             # comma-separated list of supported languages
CodePage      =                # code page to convert source file into utf-8, e.g.: windows-1252
MaxRowCount   = 100            # max number of rows to return from read parameters or output tables
MaxRunHistory = 1000           # max number of model runs to keep in run list history
DoubleFormat  = %.15g          # format to convert float or double value to string, e.g. %.15g

; Log settings:
; log can be enabled/disabled for 3 independent streams:
;   console             - standard output
;   "current" log file  - log file with specified name, overwritten on every model run
;   "stamped" log file  - log file with unique name, created for every model run
;
; "stamped" name produced from "current" name by adding time-stamp and/or process-id-stamp, i.e.:
;   oms.log => oms.2012_08_17_16_04_59_148.123456.log
#
; LogUseDailyStamp creates new log file every day
; by default LogUseDailyStamp:
;   = false if log file disabled (default)
;   = false if "stamped" log file enabled
;   = true  if log file enabled and "stamped" log file disabled
;
[OpenM]
LogToConsole = true      # if true then log to standard output
LogToFile = false        # if true then log to file
LogFilePath = oms.log    # log file path, default = current/dir/exeName.log
LogUseTimeStamp = false  # if true then use time-stamp in log file name
LogUsePidStamp = false   # if true then use pid-stamp in log file name
LogUseDailyStamp = false # if true then use daily-stamp in log file name
LogSql = false           # if true then log sql statements into log file
```

Oms directory structure: user home and jobs directories

Following directory structure expected by default:

```
./        -> oms "root" directory, by default it is current directory
    html/    -> web-UI directory with HTML, js, css, images...
    etc/     -> config files directory, contain template(s) to run models
    log/     -> recommended log files directory
    models/
          bin/  -> default model.exe and model.sqlite directory
          log/  -> default directory for models run log files
```

If you don't don't want web-UI or don't have `html` directory then start oms as:

```
oms -oms.ApiOnly
```

You can explicitly specify oms log files location, models and models log directory, e.g.:

```
oms -oms.ModelDir /my-models -oms.ModelLogDir /my-models-log
```

If you want to use log file and no console messages:

```
oms -OpenM.LogToConsole=false -OpenM.LogToFile
oms -OpenM.LogToConsole=false -OpenM.LogFilePath log/oms.log
```

If you want to use "daily" log files:

```
oms -OpenM.LogUseDailyStamp -OpenM.LogToFile
oms -OpenM.LogUseDailyStamp -OpenM.LogFilePath log/oms.log
```

User home directory

You can enable user home directory to store home directory for user personal settings, downloads of model model run results or upload input scenarios:

```
oms -oms.HomeDir models/home -oms.AllowDownload -oms.AllowUpload
```

Above command assume directory structure with `home`, `download` and `upload` sub-folders of `models`:

```
./        -> oms "root" directory, by default it is current directory
    html/    -> web-UI directory with HTML, js, css, images...
    etc/     -> config files directory, contain template(s) to run models
    log/     -> recommended log files directory
    models/
          bin/  -> default model.exe and model.sqlite directory
          log/  -> default directory for models run log files
          home/ -> user personal home directory
              io/download  -> user directory for download files
            io/upload    -> user directory to upload files
```

Note: openM++ `dbcopy` utility is required for download and upload, it must be located in the same directory where `oms` executable is.

Model run jobs directory structure

If you want to have model runs queue, or using openM++ in cloud and want automatically scale up and down cloud resources, e.g. start and stop virtual machines for model runs then start `oms` with job control option:

```
oms -oms.JobDir job
```

Following directory structure expected:

```
./        -> oms "root" directory, by default it is current directory
    html/    -> web-UI directory with HTML, js, css, images...
    etc/     -> config files directory, contain template(s) to run models
    log/     -> recommended log files directory
    models/
          bin/  -> default model.exe and model.sqlite directory
          log/  -> default directory for models run log files
            home/ -> user personal home directory
                io/download  -> user directory for download files
              io/upload    -> user directory to upload files
    job/  -> model run jobs control directory
          job.ini   -> (optional) job control settings
          active/   -> active model run state files
          history/  -> model run history files
          queue/    -> model run queue files
          state/    -> jobs state and computational servers state files
               jobs.queue.paused  -> if such file exists then jobs queue is paused
```

Please visit following page to find out how to use oms in cloud and manage model runs queue.

Oms: openM++ web-service API

Web-service methods arguments

```
:model   - model digest or model name
:lang    - language code
:run     - model run digest, run stamp or run name, modeling task run stamp or task run name
:set     - name of workset (input set of model parameters)
:profile - profile name
:task    - modeling task
```

See more details at: Arguments of web-service methods.

GET Model Metadata

GET model list

```
GET /api/model-list
```

GET model list including text (description and notes)

```
GET /api/model-list/text
GET /api/model-list/text/lang/:lang
```

GET model definition metadata

```
GET /api/model/:model
```

GET model metadata including text (description and notes)

```
GET /api/model/:model/text
GET /api/model/:model/text/lang/:lang
```

GET model metadata including text in all languages

```
GET /api/model/:model/text/all
```

GET Model Extras

GET model languages

```
GET /api/model/:model/lang-list
```

GET model language-specific strings

```
GET /api/model/:model/word-list
GET /api/model/:model/word-list/lang/:lang
```

GET model profile

```
GET /api/model/:model/profile/:profile
```

GET list of profiles

```
GET /api/model/:model/profile-list
```

GET Model Run results metadata

GET list of model runs

```
GET /api/model/:model/run-list
```

GET list of model runs including text (description and notes)

```
GET /api/model/:model/run-list/text
GET /api/model/:model/run-list/text/lang/:lang
```

GET status of model run

```
GET /api/model/:model/run/:run/status
```

GET status of model run list

```
GET /api/model/:model/run/:run/status/list
```

GET status of first model run

```
GET /api/model/:model/run/status/first
```

GET status of last model run

```
GET /api/model/:model/run/status/last
```

GET status of last completed model run

```
GET /api/model/:model/run/status/last-completed
```

GET model run metadata and status

```
GET /api/model/:model/run/:run
```

GET model run including text (description and notes)

```
GET /api/model/:model/run/:run/text
GET /api/model/:model/run/:run/text/lang/:lang
```

GET model run including text in all languages

```
GET /api/model/:model/run/:run/text/all
```

GET Model Workset metadata: set of input parameters

GET list of model worksets

```
GET /api/model/:model/workset-list
```

GET list of model worksets including text (description and notes)

```
GET /api/model/:model/workset-list/text
GET /api/model/:model/workset-list/text/lang/:lang
```

GET workset status

```
GET /api/model/:model/workset/:set/status
GET /api/model/:model/workset/:set
```

GET model default workset status

```
GET /api/model/:model/workset/status/default
```

GET workset including text (description and notes)

```
GET /api/model/:model/workset/:set/text
GET /api/model/:model/workset/:set/text/lang/:lang
```

GET workset including text in all languages

```
GET /api/model/:model/workset/:set/text/all
```

Read Parameters, Output Tables or Microdata values

Read parameter values from workset

```
POST /api/model/:model/workset/:set/parameter/value
```

Read parameter values from workset (enum id's)

```
POST /api/model/:model/workset/:set/parameter/value-id
```

Read parameter values from model run

```
POST /api/model/:model/run/:run/parameter/value
```

Read parameter values from model run (enum id's)

```
POST /api/model/:model/run/:run/parameter/value-id
```

Read output table values from model run

```
POST /api/model/:model/run/:run/table/value
```

Read output table values from model run (enum id's)

```
POST /api/model/:model/run/:run/table/value-id
```

Read microdata values from model run

```
POST /api/model/:model/run/:run/microdata/value
```

Read microdata values from model run (enum id's)

```
POST /api/model/:model/run/:run/microdata/value-id
```

GET Parameters, Output Tables or Microdata values

GET parameter values from workset

```
GET /api/model/:model/workset/:set/parameter/:name/value
GET /api/model/:model/workset/:set/parameter/:name/value/start/:start
GET /api/model/:model/workset/:set/parameter/:name/value/start/:start/count/:count
```

GET parameter values from model run

```
GET /api/model/:model/run/:run/parameter/:name/value
GET /api/model/:model/run/:run/parameter/:name/value/start/:start
GET /api/model/:model/run/:run/parameter/:name/value/start/:start/count/:count
```

GET output table expression(s) from model run

```
GET /api/model/:model/run/:run/table/:name/expr
GET /api/model/:model/run/:run/table/:name/expr/start/:start
GET /api/model/:model/run/:run/table/:name/expr/start/:start/count/:count
```

GET output table accumulator(s) from model run

```
GET /api/model/:model/run/:run/table/:name/acc
GET /api/model/:model/run/:run/table/:name/acc/start/:start
GET /api/model/:model/run/:run/table/:name/acc/start/:start/count/:count
```

GET output table all accumulators from model run

```
GET /api/model/:model/run/:run/table/:name/all-acc
GET /api/model/:model/run/:run/table/:name/all-acc/start/:start
GET /api/model/:model/run/:run/table/:name/all-acc/start/:start/count/:count
```

GET microdata values from model run

```
GET /api/model/:model/run/:run/microdata/:name/value
GET /api/model/:model/run/:run/microdata/:name/value/start/:start
GET /api/model/:model/run/:run/microdata/:name/value/start/:start/count/:count
```

GET Parameters, Output Tables or Microdata values as CSV

GET csv parameter values from workset

```
GET /api/model/:model/workset/:set/parameter/:name/csv
GET /api/model/:model/workset/:set/parameter/:name/csv-bom
```

GET csv parameter values from workset (enum id's)

```
GET /api/model/:model/workset/:set/parameter/:name/csv-id
GET /api/model/:model/workset/:set/parameter/:name/csv-id-bom
```

GET csv parameter values from model run

```
GET /api/model/:model/run/:run/parameter/:name/csv
GET /api/model/:model/run/:run/parameter/:name/csv-bom
```

GET csv parameter values from model run (enum id's)

```
GET /api/model/:model/run/:run/parameter/:name/csv-id
GET /api/model/:model/run/:run/parameter/:name/csv-id-bom
```

GET csv output table expressions from model run

```
GET /api/model/:model/run/:run/table/:name/expr/csv
GET /api/model/:model/run/:run/table/:name/expr/csv-bom
```

GET csv output table expressions from model run (enum id's)

```
GET /api/model/:model/run/:run/table/:name/expr/csv-id
GET /api/model/:model/run/:run/table/:name/expr/csv-id-bom
```

GET csv output table accumulators from model run

```
GET /api/model/:model/run/:run/table/:name/acc/csv
GET /api/model/:model/run/:run/table/:name/acc/csv-bom
```

GET csv output table accumulators from model run (enum id's)

```
GET /api/model/:model/run/:run/table/:name/acc/csv-id
GET /api/model/:model/run/:run/table/:name/acc/csv-id-bom
```

GET csv output table all accumulators from model run

```
GET /api/model/:model/run/:run/table/:name/all-acc/csv
GET /api/model/:model/run/:run/table/:name/all-acc/csv-bom
```

GET csv output table all accumulators from model run (enum id's)

```
GET /api/model/:model/run/:run/table/:name/all-acc/csv-id
GET /api/model/:model/run/:run/table/:name/all-acc/csv-id-bom
```

GET csv microdata values from model run

```
GET /api/model/:model/run/:run/microdata/:name/csv
GET /api/model/:model/run/:run/microdata/:name/csv-bom
```

GET csv microdata values from model run (enum id's)

```
GET /api/model/:model/run/:run/microdata/:name/csv-id
GET /api/model/:model/run/:run/microdata/:name/csv-id-bom
```

GET Modeling Task metadata and task run history

GET list of modeling tasks

```
GET /api/model/:model/task-list
```

GET list of modeling tasks including text (description and notes)

```
GET /api/model/:model/task-list/text
GET /api/model/:model/task-list/text/lang/:lang
```

GET modeling task input worksets

```
GET /api/model/:model/task/:task/sets
```

GET modeling task run history

```
GET /api/model/:model/task/:task/runs
```

GET status of modeling task run

```
GET /api/model/:model/task/:task/run-status/run/:run
```

GET status of modeling task run list

```
GET /api/model/:model/task/:task/run-status/list/:run
```

GET status of modeling task first run

```
GET /api/model/:model/task/:task/run-status/first
```

GET status of modeling task last run

```
GET /api/model/:model/task/:task/run-status/last
```

GET status of modeling task last completed run

```
GET /api/model/:model/task/:task/run-status/last-completed
```

GET modeling task including text (description and notes)

```
GET /api/model/:model/task/:task/text
GET /api/model/:model/task/:task/text/lang/:lang
```

GET modeling task text in all languages

```
GET /api/model/:model/task/:task/text/all
```

Update Model Profile: set of key-value options

PATCH create or replace profile

```
PATCH /api/model/:model/profile
```

DELETE profile

```
DELETE /api/model/:model/profile/:profile
```

POST create or replace profile option

```
POST /api/model/:model/profile/:profile/key/:key/value/:value
```

DELETE profile option

```
DELETE /api/model/:model/profile/:profile/key/:key
```

Update Model Workset: set of input parameters

POST update workset read-only status

```
POST /api/model/:model/workset/:set/readonly/:readonly
```

PUT create new workset

```
PUT  /api/workset-create
```

PUT create or replace workset

```
PUT /api/workset-replace
```

PATCH create or merge workset

```
PATCH /api/workset-merge
```

DELETE workset

```
DELETE /api/model/:model/workset/:set
```

DELETE parameter from workset

```
DELETE /api/model/:model/workset/:set/parameter/:name
```

PATCH update workset parameter values

```
PATCH /api/model/:model/workset/:set/parameter/:name/new/value
```

PATCH update workset parameter values (enum id's)

```
PATCH /api/model/:model/workset/:set/parameter/:name/new/value-id
```

PATCH update workset parameter(s) value notes

```
PATCH /api/model/:model/workset/:set/parameter-text
```

PUT copy parameter from model run into workset

```
PUT /api/model/:model/workset/:set/copy/parameter/:name/from-run/:run
```

PATCH merge parameter from model run into workset

```
PATCH /api/model/:model/workset/:set/merge/parameter/:name/from-run/:run
```

PUT copy parameter from workset to another

```
PUT /api/model/:model/workset/:set/copy/parameter/:name/from-workset/:from-set
```

PATCH merge parameter from workset to another

```
PATCH /api/model/:model/workset/:set/merge/parameter/:name/from-workset/:from-set
```

Update Model Runs

PATCH update model run text (description and notes)

```
PATCH /api/run/text
```

DELETE model run

```
DELETE /api/model/:model/run/:run
```

PATCH update run parameter(s) value notes

```
PATCH /api/model/:model/run/:run/parameter-text
```

Update Modeling Tasks

PUT create or replace modeling task

```
PUT  /api/task-new
```

PATCH create or update modeling task

```
PATCH /api/task
```

DELETE modeling task

```
DELETE /api/model/:model/task/:task
```

Run Models: run models and monitor progress

POST a request to run the model

```
POST /api/run
```

GET state of current model run

```
GET /api/run/log/model/:model/stamp/:stamp
GET /api/run/log/model/:model/stamp/:stamp/start/:start/count/:count
```

PUT stop model run

```
PUT /api/run/stop/model/:model/stamp/:stamp
```

Download model, model run results or input parameters

GET download log file

```
GET /api/download/log/file/:name
```

GET all download log files for the model

```
GET /api/download/log/model/:model
```

GET all download log files

```
GET /api/download/log/all
```

GET download files tree

```
GET /api/download/file-tree/:folder
```

POST initiate model download

```
POST /api/download/model/:model
```

POST initiate model run download

```
POST /api/download/model/:model/run/:run
```

POST initiate model workset download

```
POST /api/download/model/:model/workset/:set
```

DELETE download files

```
DELETE /api/download/delete/:folder
```

DELETE download files asynchronously

```
DELETE /api/download/start/delete/:folder
```

Upload model runs or worksets

GET upload log file

```
GET /api/upload/log/file/:name
```

GET all upload log files for the model

```
GET /api/upload/log/model/:model
```

GET all upload log files

```
GET /api/upload/log/all
```

GET upload files tree

```
GET /api/upload/file-tree/:folder
```

POST initiate model run upload

```
POST /api/upload/model/:model/run
POST /api/upload/model/:model/run/:run
```

POST initiate workset upload

```
POST /api/upload/model/:model/workset
POST /api/upload/model/:model/workset/:set
```

DELETE upload files

```
DELETE /api/upload/delete/:folder
```

DELETE upload files asynchronously

```
DELETE /api/upload/start/delete/:folder
```

User: manage user settings and data

GET user views for the model

```
GET /api/user/view/model/:model
```

PUT user views for the model

```
PUT /api/user/view/model/:model
```

DELETE user views for the model

```
DELETE /api/user/view/model/:model
```

Model run jobs and service state

GET web-service configuration

```
GET /api/service/config
```

GET web-service state

```
GET /api/service/state
```

GET state of active model run job

```
GET /api/service/job/active/:job
```

GET state of model run job from queue

```
GET /api/service/job/queue/:job
```

GET state of model run job from history

```
GET /api/service/job/history/:job
```

PUT model run job into other queue position

```
PUT /api/service/job/move/:pos/:job
```

DELETE state of model run job from history

```
DELETE /api/service/job/delete/history/:job
```

Administrative: manage web-service state

POST a request to refresh models catalog

```
POST /api/admin/all-models/refresh
```

POST a request to close models catalog

```
POST /api/admin/all-models/close
```

POST a request to pause model run queue

```
POST /api/admin/jobs-pause/:pause
```

PUT a request to shutdown web-service

```
PUT /api/admin/shutdown
```

GET model list

Get list of the models.

Method:

```
GET /api/model-list
```

Call example:

```
http://localhost:4040/api/model-list
```

Return example: *This is a beta version and may change in the future.*


```
[
  {
    "ModelId": 101,
    "Name": "IDMM",
    "Digest": "0f76e04fb52de763f836c7b026c00f80",
    "Type": 1,
    "Version": "2.0.0.0",
    "CreateDateTime": "2017-12-19 15:19:57.0747",
    "DefaultLangCode": "EN"
  },
  {
    "ModelId": 101,
    "Name": "NewCaseBased",
    "Digest": "649f17f26d67c37b78dde94f79772445",
    "Type": 0,
    "Version": "1.0.0.0",
    "CreateDateTime": "2017-12-19 15:21:14.0232",
    "DefaultLangCode": "EN"
  },
  {
    "ModelId": 101,
    "Name": "NewTimeBased",
    "Digest": "0ceaa8fbc0b762c5cb287a4910ede8f7",
    "Type": 1,
    "Version": "1.0.1.0",
    "CreateDateTime": "2017-12-19 15:21:47.0408",
    "DefaultLangCode": "EN"
  },
  {
    "ModelId": 1,
    "Name": "modelOne",
    "Digest": "_201208171604590148_",
    "Type": 0,
    "Version": "1.0",
    "CreateDateTime": "2012-08-17 16:04:59.0148",
    "DefaultLangCode": "EN"
  }
]
```

GET model list including text (description and notes)

Get model list including text (description and notes).

Methods:

```
GET /api/model-list/text
GET /api/model-list/text/lang/:lang
```

Arguments:

```
:lang - (optional) language code
```

If optional `lang` argument specified then result in that language else in browser language or model default. If no such language exist then result in model default language or can be empty.

Call examples:

```
http://localhost:4040/api/model-list/text
http://localhost:4040/api/model-list/text/lang/en
```

Return example: *This is a beta version and may change in the future.*

```
[
  {
    "Model": {
      "ModelId": 101,
      "Name": "IDMM",
      "Digest": "0f76e04fb52de763f836c7b026c00f80",
      "Type": 1,
      "Version": "2.0.0.0",
      "CreateDateTime": "2017-12-19 15:19:57.0747",
      "DefaultLangCode": "EN"
    },
    "DescrNote": {
      "LangCode": "EN",
      "Descr": "model",
      "Note": ""
    }
  },
  {
    "Model": {
      "ModelId": 101,
      "Name": "NewCaseBased",
      "Digest": "649f17f26d67c37b78dde94f79772445",
      "Type": 0,
      "Version": "1.0.0.0",
      "CreateDateTime": "2017-12-19 15:21:14.0232",
      "DefaultLangCode": "EN"
    },
    "DescrNote": {
      "LangCode": "EN",
      "Descr": "Simple case-based model",
      "Note": "This model can serve as a starting point for more complex case-based models."
    }
  },
  {
    "Model": {
      "ModelId": 101,
      "Name": "NewTimeBased",
      "Digest": "0ceaa8fbc0b762c5cb287a4910ede8f7",
      "Type": 1,
      "Version": "1.0.1.0",
      "CreateDateTime": "2017-12-19 15:21:47.0408",
      "DefaultLangCode": "EN"
    },
    "DescrNote": {
      "LangCode": "EN",
      "Descr": "Simple time-based model",
      "Note": "This model can serve as a starting point for more complex time-based models."
    }
  },
  {
    "Model": {
      "ModelId": 1,
      "Name": "modelOne",
      "Digest": "_201208171604590148_",
      "Type": 0,
      "Version": "1.0",
      "CreateDateTime": "2012-08-17 16:04:59.0148",
      "DefaultLangCode": "EN"
    },
    "DescrNote": {
      "LangCode": "EN",
      "Descr": "First model",
      "Note": "First model: openM++ development test model"
    }
  }
]
```

GET model definition metadata

Get model definition: language-neutral part of model metadata.

Methods:

```
GET /api/model/:model
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

Call examples:

```
http://localhost:4040/api/model/modelOne
http://localhost:4040/api/model/649f17f26d67c37b78dde94f79772445
```

Return example:


```
{
  "Model": {
    "ModelId": 1,
    "Name": "modelOne",
    "Digest": "_201208171604590148_",
    "Type": 0,
    "Version": "1.0",
    "CreateDateTime": "2012-08-17 16:04:59.148",
    "DefaultLangCode": "EN"
  },
  "Type": [
    {
      "ModelId": 1,
      "TypeId": 4,
      "TypeHid": 4,
      "Name": "int",
      "Digest": "_int_",
      "DicId": 0,
      "TotalEnumId": 1,
      "Enum": null
    },
    {
      "ModelId": 1,
      "TypeId": 7,
      "TypeHid": 7,
      "Name": "bool",
      "Digest": "_bool_",
      "DicId": 1,
      "TotalEnumId": 2,
      "Enum": [
        {
          "ModelId": 1,
          "TypeId": 7,
          "EnumId": 0,
          "Name": "false"
        },
        {
          "ModelId": 1,
          "TypeId": 7,
          "EnumId": 1,
          "Name": "true"
        }
      ]
    },
    {
      "ModelId": 1,
      "TypeId": 14,
      "TypeHid": 14,
      "Name": "double",
      "Digest": "_double_",
      "DicId": 0,
      "TotalEnumId": 1,
      "Enum": null
    },
    {
      "ModelId": 1,
```

```
    "ModelId": 1,
    "TypeId": 21,
    "TypeHid": 21,
    "Name": "file",
    "Digest": "_file_",
    "DicId": 0,
    "TotalEnumId": 1,
    "Enum": null
  },
  {
    "ModelId": 1,
    "TypeId": 101,
    "TypeHid": 96,
    "Name": "age",
    "Digest": "_20128171604590121",
    "DicId": 2,
    "TotalEnumId": 500,
    "Enum": [
      {
        "ModelId": 1,
        "TypeId": 101,
        "EnumId": 10,
        "Name": "10-20"
      },
      {
        "ModelId": 1,
        "TypeId": 101,
        "EnumId": 20,
        "Name": "20-30"
      },
      {
        "ModelId": 1,
        "TypeId": 101,
        "EnumId": 30,
        "Name": "30-40"
      },
      {
        "ModelId": 1,
        "TypeId": 101,
        "EnumId": 40,
        "Name": "40+"
      }
    ]
  },
  {
    "ModelId": 1,
    "TypeId": 102,
    "TypeHid": 97,
    "Name": "sex",
    "Digest": "_20128171604590122",
    "DicId": 2,
    "TotalEnumId": 800,
    "Enum": [
      {
        "ModelId": 1,
        "TypeId": 102,
        "EnumId": 0,
        "Name": "M"
      },
      {
        "ModelId": 1,
        "TypeId": 102,
        "EnumId": 1,
        "Name": "F"
      }
    ]
  },
  {
    "ModelId": 1,
    "TypeId": 103,
    "TypeHid": 98,
    "Name": "salary",
    "Digest": "_20128171604590123",
    "DicId": 2,
    "TotalEnumId": 400,
    "Enum": [
      {
        "ModelId": 1,
        "TypeId": 103,
        "EnumId": 100,
        "Name": "L"
      },
      {
        "ModelId": 1,
        "TypeId": 103,
        "EnumId": 200,
        "Name": "M"
      },
      {
```

```
      {
        "ModelId": 1,
        "TypeId": 103,
        "EnumId": 300,
        "Name": "H"
      }
    ]
  },
  {
    "ModelId": 1,
    "TypeId": 104,
    "TypeHid": 99,
    "Name": "full",
    "Digest": "_20128171604590124",
    "DicId": 2,
    "TotalEnumId": 44,
    "Enum": [
      {
        "ModelId": 1,
        "TypeId": 104,
        "EnumId": 22,
        "Name": "Full"
      },
      {
        "ModelId": 1,
        "TypeId": 104,
        "EnumId": 33,
        "Name": "Part"
      }
    ]
  }
],
"Param": [
  {
    "ModelId": 1,
    "ParamId": 0,
    "ParamHid": 44,
    "Name": "ageSex",
    "Digest": "_20128171604590131",
    "Rank": 2,
    "TypeId": 14,
    "IsExtendable": true,
    "IsHidden": false,
    "NumCumulated": 0,
    "DbRunTable": "ageSex_p_2012817",
    "DbSetTable": "ageSex_w_2012817",
    "ImportDigest": "_i0128171604590131",
    "Dim": [
      {
        "ModelId": 1,
        "ParamId": 0,
        "DimId": 0,
        "Name": "dim0",
        "TypeId": 101
      },
      {
        "ModelId": 1,
        "ParamId": 0,
        "DimId": 1,
        "Name": "dim1",
        "TypeId": 102
      }
    ],
    "Import": [
      {
        "ModelId": 1,
        "ParamId": 0,
        "FromName": "ageSexIncome",
        "FromModel": "modelOne",
        "IsSampleDim": false
      }
    ]
  },
  {
    "ModelId": 1,
    "ParamId": 1,
    "ParamHid": 45,
    "Name": "salaryAge",
    "Digest": "_20128171604590132",
    "Rank": 2,
    "TypeId": 4,
    "IsExtendable": false,
    "IsHidden": false,
    "NumCumulated": 0,
    "DbRunTable": "salaryAge_p_2012818",
    "DbSetTable": "salaryAge_w_2012818",
    "ImportDigest": "_i0128171604590132",
    "Dim": [
```

```
      {
        "ModelId": 1,
        "ParamId": 1,
        "DimId": 0,
        "Name": "dim0",
        "TypeId": 103
      },
      {
        "ModelId": 1,
        "ParamId": 1,
        "DimId": 1,
        "Name": "dim1",
        "TypeId": 101
      }
    ],
    "Import": [
      {
        "ModelId": 1,
        "ParamId": 1,
        "FromName": "salaryAge",
        "FromModel": "modelOne",
        "IsSampleDim": false
      }
    ]
  },
  {
    "ModelId": 1,
    "ParamId": 2,
    "ParamHid": 46,
    "Name": "StartingSeed",
    "Digest": "_20128171604590133",
    "Rank": 0,
    "TypeId": 4,
    "IsExtendable": false,
    "IsHidden": false,
    "NumCumulated": 0,
    "DbRunTable": "StartingSeed_p_2012819",
    "DbSetTable": "StartingSeed_w_2012819",
    "ImportDigest": "_i0128171604590133",
    "Dim": null,
    "Import": [
      {
        "ModelId": 1,
        "ParamId": 2,
        "FromName": "StartingSeed",
        "FromModel": "modelOne",
        "IsSampleDim": false
      }
    ]
  },
  {
    "ModelId": 1,
    "ParamId": 3,
    "ParamHid": 47,
    "Name": "salaryFull",
    "Digest": "_20128171604590134",
    "Rank": 1,
    "TypeId": 104,
    "IsExtendable": false,
    "IsHidden": false,
    "NumCumulated": 0,
    "DbRunTable": "salaryFull_p_2012812",
    "DbSetTable": "salaryFull_w_2012812",
    "ImportDigest": "_i0128171604590134",
    "Dim": [
      {
        "ModelId": 1,
        "ParamId": 3,
        "DimId": 0,
        "Name": "dim0",
        "TypeId": 103
      }
    ],
    "Import": null
  },
  {
    "ModelId": 1,
    "ParamId": 4,
    "ParamHid": 48,
    "Name": "baseSalary",
    "Digest": "_20128171604590135",
    "Rank": 0,
    "TypeId": 104,
    "IsExtendable": false,
    "IsHidden": false,
    "NumCumulated": 0,
    "DbRunTable": "baseSalary_p_2012811",
    "DbSetTable": "baseSalary_w_2012811",
```

```
      "DbSetTable": "baseSalary_w_2012811",
      "ImportDigest": "_i0128171604590135",
      "Dim": null,
      "Import": null
    },
    {
      "ModelId": 1,
      "ParamId": 5,
      "ParamHid": 49,
      "Name": "filePath",
      "Digest": "_20128171604590136",
      "Rank": 0,
      "TypeId": 21,
      "IsExtendable": false,
      "IsHidden": false,
      "NumCumulated": 0,
      "DbRunTable": "filePath_p_2012814",
      "DbSetTable": "filePath_w_2012814",
      "ImportDigest": "_i0128171604590136",
      "Dim": null,
      "Import": null
    },
    {
      "ModelId": 1,
      "ParamId": 6,
      "ParamHid": 50,
      "Name": "isOldAge",
      "Digest": "_20128171604590137",
      "Rank": 1,
      "TypeId": 7,
      "IsExtendable": false,
      "IsHidden": false,
      "NumCumulated": 0,
      "DbRunTable": "isOldAge_p_2012815",
      "DbSetTable": "isOldAge_w_2012815",
      "ImportDigest": "_i0128171604590137",
      "Dim": [
        {
          "ModelId": 1,
          "ParamId": 6,
          "DimId": 0,
          "Name": "dim0",
          "TypeId": 101
        }
      ],
      "Import": null
    }
  ],
  "Table": [
    {
      "ModelId": 1,
      "TableId": 0,
      "TableHid": 82,
      "Name": "salarySex",
      "Digest": "_20128171604590182",
      "IsUser": false,
      "Rank": 2,
      "IsSparse": true,
      "DbExprTable": "salarySex_v_2012882",
      "DbAccTable": "salarySex_a_2012882",
      "DbAccAllView": "salarySex_d_2012882",
      "ExprPos": 1,
      "IsHidden": false,
      "ImportDigest": "_i0128171604590182",
      "Dim": [
        {
          "ModelId": 1,
          "TableId": 0,
          "DimId": 0,
          "Name": "dim0",
          "TypeId": 103,
          "IsTotal": false,
          "DimSize": 3
        },
        {
          "ModelId": 1,
          "TableId": 0,
          "DimId": 1,
          "Name": "dim1",
          "TypeId": 102,
          "IsTotal": true,
          "DimSize": 3
        }
      ],
      "Acc": [
        {
          "ModelId": 1,
          "TableId": 0,
```

```
          "AccId": 0,
          "Name": "acc0",
          "IsDerived": false,
          "SrcAcc": "value_sum()",
          "AccSql": "A.acc_value"
        },
        {
          "ModelId": 1,
          "TableId": 0,
          "AccId": 1,
          "Name": "acc1",
          "IsDerived": false,
          "SrcAcc": "value_count()",
          "AccSql": "SELECT A1.acc_value FROM salarySex_a_2012882 A1 WHERE A1.run_id = A.run_id AND A1.sub_id = A.sub_id AND A1.dim0 = A.dim0 AND A1.dim1 = A.dim1 AND A1.acc_id = 1"
        },
        {
          "ModelId": 1,
          "TableId": 0,
          "AccId": 2,
          "Name": "acc2",
          "IsDerived": true,
          "SrcAcc": "acc0 + acc1",
          "AccSql": "(A.acc_value) + (SELECT A1.acc_value FROM salarySex_a_2012882 A1 WHERE A1.run_id = A.run_id AND A1.sub_id = A.sub_id AND A1.dim0 = A.dim0 AND A1.dim1 = A.dim1 AND A1.acc_id = 1)"
        }
      ],
      "Expr": [
        {
          "ModelId": 1,
          "TableId": 0,
          "ExprId": 0,
          "Name": "expr0",
          "Decimals": 4,
          "SrcExpr": "OM_AVG(acc0)",
          "ExprSql": "SELECT M1.run_id, M1.dim0, M1.dim1, AVG(M1.acc_value) AS expr0 FROM salarySex_a_2012882 M1 WHERE M1.acc_id = 0 GROUP BY M1.run_id, M1.dim0, M1.dim1"
        },
        {
          "ModelId": 1,
          "TableId": 0,
          "ExprId": 1,
          "Name": "expr1",
          "Decimals": 4,
          "SrcExpr": "OM_SUM(acc1)",
          "ExprSql": "SELECT M1.run_id, M1.dim0, M1.dim1, SUM(M1.acc_value) AS expr1 FROM salarySex_a_2012882 M1 WHERE M1.acc_id = 1 GROUP BY M1.run_id, M1.dim0, M1.dim1"
        },
        {
          "ModelId": 1,
          "TableId": 0,
          "ExprId": 2,
          "Name": "expr2",
          "Decimals": 2,
          "SrcExpr": "OM_MIN(acc0)",
          "ExprSql": "SELECT M1.run_id, M1.dim0, M1.dim1, MIN(M1.acc_value) AS expr2 FROM salarySex_a_2012882 M1 WHERE M1.acc_id = 0 GROUP BY M1.run_id, M1.dim0, M1.dim1"
        },
        {
          "ModelId": 1,
          "TableId": 0,
          "ExprId": 3,
          "Name": "expr3",
          "Decimals": 3,
          "SrcExpr": "OM_AVG(acc0 * acc1)",
          "ExprSql": "SELECT M1.run_id, M1.dim0, M1.dim1, AVG(M1.acc_value * A1.acc1) AS expr3 FROM salarySex_a_2012882 M1 INNER JOIN (SELECT run_id, dim0, dim1, sub_id, acc_value AS acc1 FROM salarySex_a_2012882 WHERE acc_id = 1) A1 ON (A1.run_id = M1.run_id AND A1.dim0 = M1.dim0 AND A1.dim1 = M1.dim1 AND A1.sub_id = M1.sub_id) WHERE M1.acc_id = 0 GROUP BY M1.run_id, M1.dim0, M1.dim1"
        }
      ]
    },
    {
      "ModelId": 1,
      "TableId": 1,
      "TableHid": 83,
      "Name": "fullAgeSalary",
      "Digest": "_20128171604590183",
      "IsUser": false,
      "Rank": 3,
      "IsSparse": false,
      "DbExprTable": "fullAgeSalary_v_2012883",
      "DbAccTable": "fullAgeSalary_a_2012883",
      "DbAccAllView": "fullAgeSalary_d_2012883",
      "ExprPos": 1,
      "IsHidden": false,
      "ImportDigest": "_i0128171604590183",
      "Dim": [
        {
```

```
        {
          "ModelId": 1,
          "TableId": 1,
          "DimId": 0,
          "Name": "dim0",
          "TypeId": 104,
          "IsTotal": false,
          "DimSize": 2
        },
        {
          "ModelId": 1,
          "TableId": 1,
          "DimId": 1,
          "Name": "dim1",
          "TypeId": 101,
          "IsTotal": true,
          "DimSize": 5
        },
        {
          "ModelId": 1,
          "TableId": 1,
          "DimId": 2,
          "Name": "dim2",
          "TypeId": 103,
          "IsTotal": false,
          "DimSize": 3
        }
      ],
      "Acc": [
        {
          "ModelId": 1,
          "TableId": 1,
          "AccId": 0,
          "Name": "acc0",
          "IsDerived": false,
          "SrcAcc": "raw_value()",
          "AccSql": "A.acc_value"
        }
      ],
      "Expr": [
        {
          "ModelId": 1,
          "TableId": 1,
          "ExprId": 0,
          "Name": "expr0",
          "Decimals": 2,
          "SrcExpr": "OM_AVG(acc0)",
          "ExprSql": "SELECT M1.run_id, M1.dim0, M1.dim1, M1.dim2, AVG(M1.acc_value) AS expr0 FROM fullAgeSalary_a_2012883 M1 WHERE M1.acc_id = 0 GROUP BY M1.run_id, M1.dim0, M1.dim1, M1.dim2"
        }
      ]
    },
    {
      "ModelId": 1,
      "TableId": 2,
      "TableHid": 84,
      "Name": "ageSexIncome",
      "Digest": "_20128171604590184",
      "IsUser": false,
      "Rank": 2,
      "IsSparse": false,
      "DbExprTable": "ageSexIncome_v_2012884",
      "DbAccTable": "ageSexIncome_a_2012884",
      "DbAccAllView": "ageSexIncome_d_2012884",
      "ExprPos": 0,
      "IsHidden": false,
      "ImportDigest": "_i0128171604590131",
      "Dim": [
        {
          "ModelId": 1,
          "TableId": 2,
          "DimId": 0,
          "Name": "dim0",
          "TypeId": 101,
          "IsTotal": false,
          "DimSize": 4
        },
        {
          "ModelId": 1,
          "TableId": 2,
          "DimId": 1,
          "Name": "dim1",
          "TypeId": 102,
          "IsTotal": false,
          "DimSize": 2
        }
      ],
      "Acc": [
        {
```

```
      {
        "ModelId": 1,
        "TableId": 2,
        "AccId": 0,
        "Name": "acc0",
        "IsDerived": false,
        "SrcAcc": "raw_value()",
        "AccSql": "A.acc_value"
      },
      {
        "ModelId": 1,
        "TableId": 2,
        "AccId": 1,
        "Name": "acc1",
        "IsDerived": false,
        "SrcAcc": "adjust_value()",
        "AccSql": "A.acc_value"
      }
    ],
    "Expr": [
      {
        "ModelId": 1,
        "TableId": 2,
        "ExprId": 0,
        "Name": "expr0",
        "Decimals": 2,
        "SrcExpr": "OM_AVG(acc0)",
        "ExprSql": "SELECT M1.run_id, M1.dim0, M1.dim1, AVG(M1.acc_value) AS expr0 FROM ageSexIncome_a_2012884 M1 WHERE M1.acc_id = 0 GROUP BY M1.run_id, M1.dim0, M1.dim1"
      },
      {
        "ModelId": 1,
        "TableId": 2,
        "ExprId": 1,
        "Name": "expr1",
        "Decimals": 3,
        "SrcExpr": "OM_AVG(acc1)",
        "ExprSql": "SELECT M1.run_id, M1.dim0, M1.dim1, AVG(M1.acc_value) AS expr1 FROM ageSexIncome_a_2012884 M1 WHERE M1.acc_id = 1 GROUP BY M1.run_id, M1.dim0, M1.dim1"
      }
    ]
  },
  {
    "ModelId": 1,
    "TableId": 3,
    "TableHid": 85,
    "Name": "seedOldAge",
    "Digest": "_20128171604590185",
    "IsUser": false,
    "Rank": 0,
    "IsSparse": false,
    "DbExprTable": "seedOldAge_v_2012885",
    "DbAccTable": "seedOldAge_a_2012885",
    "DbAccAllView": "seedOldAge_d_2012885",
    "ExprPos": 0,
    "IsHidden": false,
    "ImportDigest": "_i0128171604590185",
    "Dim": null,
    "Acc": [
      {
        "ModelId": 1,
        "TableId": 3,
        "AccId": 0,
        "Name": "acc0",
        "IsDerived": false,
        "SrcAcc": "raw_value()",
        "AccSql": "A.acc_value"
      }
    ],
    "Expr": [
      {
        "ModelId": 1,
        "TableId": 3,
        "ExprId": 0,
        "Name": "expr0",
        "Decimals": 5,
        "SrcExpr": "OM_AVG(acc0)",
        "ExprSql": "SELECT M1.run_id, AVG(M1.acc_value) AS expr0 FROM seedOldAge_a_2012885 M1 WHERE M1.acc_id = 0 GROUP BY M1.run_id"
      }
    ]
  }
 ],
 "Group": [
  {
    "ModelId": 1,
    "GroupId": 1,
    "IsParam": true,
    "Name": "AllParameters",
```

```
    "IsHidden": false,
    "GroupPc": [
      {
        "ModelId": 1,
        "GroupId": 1,
        "ChildPos": 0,
        "ChildGroupId": 2,
        "ChildLeafId": -1
      },
      {
        "ModelId": 1,
        "GroupId": 1,
        "ChildPos": 1,
        "ChildGroupId": 3,
        "ChildLeafId": -1
      },
      {
        "ModelId": 1,
        "GroupId": 1,
        "ChildPos": 2,
        "ChildGroupId": -1,
        "ChildLeafId": 2
      },
      {
        "ModelId": 1,
        "GroupId": 1,
        "ChildPos": 3,
        "ChildGroupId": -1,
        "ChildLeafId": 5
      }
    ]
  },
  {
    "ModelId": 1,
    "GroupId": 2,
    "IsParam": true,
    "Name": "AgeSexParameters",
    "IsHidden": false,
    "GroupPc": [
      {
        "ModelId": 1,
        "GroupId": 2,
        "ChildPos": 0,
        "ChildGroupId": -1,
        "ChildLeafId": 0
      },
      {
        "ModelId": 1,
        "GroupId": 2,
        "ChildPos": 1,
        "ChildGroupId": -1,
        "ChildLeafId": 1
      },
      {
        "ModelId": 1,
        "GroupId": 2,
        "ChildPos": 2,
        "ChildGroupId": -1,
        "ChildLeafId": 6
      }
    ]
  },
  {
    "ModelId": 1,
    "GroupId": 3,
    "IsParam": true,
    "Name": "SalaryParameters",
    "IsHidden": false,
    "GroupPc": [
      {
        "ModelId": 1,
        "GroupId": 3,
        "ChildPos": 0,
        "ChildGroupId": -1,
        "ChildLeafId": 1
      },
      {
        "ModelId": 1,
        "GroupId": 3,
        "ChildPos": 1,
        "ChildGroupId": -1,
        "ChildLeafId": 3
      },
      {
        "ModelId": 1,
        "GroupId": 3,
        "ChildPos": 2,
        "ChildGroupId": -1,
```

```
        "ChildGroupId": -1,
        "ChildLeafId": 4
      }
    ]
  },
  {
    "ModelId": 1,
    "GroupId": 10,
    "IsParam": false,
    "Name": "AdditionalTables",
    "IsHidden": false,
    "GroupPc": [
      {
        "ModelId": 1,
        "GroupId": 10,
        "ChildPos": 0,
        "ChildGroupId": -1,
        "ChildLeafId": 1
      },
      {
        "ModelId": 1,
        "GroupId": 10,
        "ChildPos": 1,
        "ChildGroupId": -1,
        "ChildLeafId": 2
      },
      {
        "ModelId": 1,
        "GroupId": 10,
        "ChildPos": 2,
        "ChildGroupId": -1,
        "ChildLeafId": 3
      }
    ]
  }
 ]
}
```

GET model metadata including text (description and notes)

Get model metadata including text (description and notes) in current user language.

Methods:

```
GET /api/model/:model/text
GET /api/model/:model/text/lang/:lang
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:lang - (optional) language code
```

If optional `lang` argument specified then result in that language else in browser language or model default. If no such language exist then result in model default language or can be empty.

Call examples:

```
http://localhost:4040/api/model/modelOne/text
http://localhost:4040/api/model/modelOne/text/lang/en
http://localhost:4040/api/model/_201208171604590148_/text/lang/en_CA
```

Return example:


```
{
 "Model": {
  "ModelId": 1,
  "Name": "modelOne",
  "Digest": "_201208171604590148_",
  "Type": 0,
  "Version": "1.0",
  "CreateDateTime": "2012-08-17 16:04:59.148",
  "DefaultLangCode": "EN"
 },
 "DescrNote": {
  "LangCode": "EN",
  "Descr": "First model",
  "Note": "First model: openM++ development test model"
 },
 "TypeTxt": [
  {
   "Type": {
    "ModelId": 1,
    "TypeId": 4,
    "TypeHid": 4,
    "Name": "int",
    "Digest": "_int_",
    "DicId": 0,
    "TotalEnumId": 1
   },
   "DescrNote": {
    "LangCode": "",
    "Descr": "",
    "Note": ""
   },
   "TypeEnumTxt": []
  },
  {
   "Type": {
    "ModelId": 1,
    "TypeId": 7,
    "TypeHid": 7,
    "Name": "bool",
    "Digest": "_bool_",
    "DicId": 1,
    "TotalEnumId": 2
   },
   "DescrNote": {
    "LangCode": "EN",
    "Descr": "logical type",
```

```
      "Note": ""
    },
    "TypeEnumTxt": [
      {
        "Enum": {
          "ModelId": 1,
          "TypeId": 7,
          "EnumId": 0,
          "Name": "false"
        },
        "DescrNote": {
          "LangCode": "EN",
          "Descr": "False",
          "Note": ""
        }
      },
      {
        "Enum": {
          "ModelId": 1,
          "TypeId": 7,
          "EnumId": 1,
          "Name": "true"
        },
        "DescrNote": {
          "LangCode": "EN",
          "Descr": "True",
          "Note": ""
        }
      }
    ]
  },
  {
    "Type": {
      "ModelId": 1,
      "TypeId": 14,
      "TypeHid": 14,
      "Name": "double",
      "Digest": "_double_",
      "DicId": 0,
      "TotalEnumId": 1
    },
    "DescrNote": {
      "LangCode": "",
      "Descr": "",
      "Note": ""
    },
    "TypeEnumTxt": []
  },
  {
    "Type": {
      "ModelId": 1,
      "TypeId": 21,
      "TypeHid": 21,
      "Name": "file",
      "Digest": "_file_",
      "DicId": 0,
      "TotalEnumId": 1
    },
    "DescrNote": {
      "LangCode": "",
      "Descr": "",
      "Note": ""
    },
    "TypeEnumTxt": []
  },
  {
    "Type": {
      "ModelId": 1,
      "TypeId": 101,
      "TypeHid": 96,
      "Name": "age",
      "Digest": "_20128171604590121",
      "DicId": 2,
      "TotalEnumId": 500
    },
    "DescrNote": {
      "LangCode": "EN",
      "Descr": "Age",
      "Note": ""
    },
    "TypeEnumTxt": [
      {
        "Enum": {
          "ModelId": 1,
          "TypeId": 101,
          "EnumId": 10,
          "Name": "10-20"
        },
```

```
      },
      "DescrNote": {
        "LangCode": "EN",
        "Descr": "age 10-20",
        "Note": ""
      }
    },
    {
      "Enum": {
        "ModelId": 1,
        "TypeId": 101,
        "EnumId": 20,
        "Name": "20-30"
      },
      "DescrNote": {
        "LangCode": "EN",
        "Descr": "age 20-30",
        "Note": ""
      }
    },
    {
      "Enum": {
        "ModelId": 1,
        "TypeId": 101,
        "EnumId": 30,
        "Name": "30-40"
      },
      "DescrNote": {
        "LangCode": "EN",
        "Descr": "age 30-40",
        "Note": ""
      }
    },
    {
      "Enum": {
        "ModelId": 1,
        "TypeId": 101,
        "EnumId": 40,
        "Name": "40+"
      },
      "DescrNote": {
        "LangCode": "EN",
        "Descr": "age 40+",
        "Note": ""
      }
    }
  ]
},
{
  "Type": {
    "ModelId": 1,
    "TypeId": 102,
    "TypeHid": 97,
    "Name": "sex",
    "Digest": "_20128171604590122",
    "DicId": 2,
    "TotalEnumId": 800
  },
  "DescrNote": {
    "LangCode": "EN",
    "Descr": "Sex",
    "Note": ""
  },
  "TypeEnumTxt": [
    {
      "Enum": {
        "ModelId": 1,
        "TypeId": 102,
        "EnumId": 0,
        "Name": "M"
      },
      "DescrNote": {
        "LangCode": "EN",
        "Descr": "Male",
        "Note": ""
      }
    },
    {
      "Enum": {
        "ModelId": 1,
        "TypeId": 102,
        "EnumId": 1,
        "Name": "F"
      },
      "DescrNote": {
        "LangCode": "EN",
        "Descr": "Female",
        "Note": ""
      }
```

```
      }
    }
  ]
},
{
  "Type": {
    "ModelId": 1,
    "TypeId": 103,
    "TypeHid": 98,
    "Name": "salary",
    "Digest": "_20128171604590123",
    "DicId": 2,
    "TotalEnumId": 400
  },
  "DescrNote": {
    "LangCode": "EN",
    "Descr": "Salary",
    "Note": ""
  },
  "TypeEnumTxt": [
    {
      "Enum": {
        "ModelId": 1,
        "TypeId": 103,
        "EnumId": 100,
        "Name": "L"
      },
      "DescrNote": {
        "LangCode": "EN",
        "Descr": "Low",
        "Note": ""
      }
    },
    {
      "Enum": {
        "ModelId": 1,
        "TypeId": 103,
        "EnumId": 200,
        "Name": "M"
      },
      "DescrNote": {
        "LangCode": "EN",
        "Descr": "Medium",
        "Note": ""
      }
    },
    {
      "Enum": {
        "ModelId": 1,
        "TypeId": 103,
        "EnumId": 300,
        "Name": "H"
      },
      "DescrNote": {
        "LangCode": "EN",
        "Descr": "High",
        "Note": ""
      }
    }
  ]
},
{
  "Type": {
    "ModelId": 1,
    "TypeId": 104,
    "TypeHid": 99,
    "Name": "full",
    "Digest": "_20128171604590124",
    "DicId": 2,
    "TotalEnumId": 44
  },
  "DescrNote": {
    "LangCode": "EN",
    "Descr": "Full or part time",
    "Note": ""
  },
  "TypeEnumTxt": [
    {
      "Enum": {
        "ModelId": 1,
        "TypeId": 104,
        "EnumId": 22,
        "Name": "Full"
      },
      "DescrNote": {
        "LangCode": "EN",
        "Descr": "Full-time",
        "Note": ""
```

```
        }
      },
      {
        "Enum": {
          "ModelId": 1,
          "TypeId": 104,
          "EnumId": 33,
          "Name": "Part"
        },
        "DescrNote": {
          "LangCode": "EN",
          "Descr": "Part-time",
          "Note": ""
        }
      }
    ]
  }
],
"ParamTxt": [
  {
    "Param": {
      "ModelId": 1,
      "ParamId": 0,
      "ParamHid": 44,
      "Name": "ageSex",
      "Digest": "_20128171604590131",
      "Rank": 2,
      "TypeId": 14,
      "IsExtendable": true,
      "IsHidden": false,
      "NumCumulated": 0,
      "DbRunTable": "ageSex_p_2012817",
      "DbSetTable": "ageSex_w_2012817",
      "ImportDigest": "_i0128171604590131"
    },
    "DescrNote": {
      "LangCode": "EN",
      "Descr": "Age by Sex",
      "Note": "Age by Sex note"
    },
    "ParamDimsTxt": [
      {
        "Dim": {
          "ModelId": 1,
          "ParamId": 0,
          "DimId": 0,
          "Name": "dim0",
          "TypeId": 101
        },
        "DescrNote": {
          "LangCode": "EN",
          "Descr": "Age Dim",
          "Note": "Age Dim notes"
        }
      },
      {
        "Dim": {
          "ModelId": 1,
          "ParamId": 0,
          "DimId": 1,
          "Name": "dim1",
          "TypeId": 102
        },
        "DescrNote": {
          "LangCode": "EN",
          "Descr": "Sex Dim",
          "Note": "Sex Dim notes"
        }
      }
    ]
  },
  {
    "Param": {
      "ModelId": 1,
      "ParamId": 1,
      "ParamHid": 45,
      "Name": "salaryAge",
      "Digest": "_20128171604590132",
      "Rank": 2,
      "TypeId": 4,
      "IsExtendable": false,
      "IsHidden": false,
      "NumCumulated": 0,
      "DbRunTable": "salaryAge_p_2012818",
      "DbSetTable": "salaryAge_w_2012818",
      "ImportDigest": "_i0128171604590132"
    },
    "DescrNote": {
```

```
      "DescrNote": {
        "LangCode": "EN",
        "Descr": "Salary by Age",
        "Note": "Salary by Age note"
      },
      "ParamDimsTxt": [
        {
          "Dim": {
            "ModelId": 1,
            "ParamId": 1,
            "DimId": 0,
            "Name": "dim0",
            "TypeId": 103
          },
          "DescrNote": {
            "LangCode": "",
            "Descr": "",
            "Note": ""
          }
        },
        {
          "Dim": {
            "ModelId": 1,
            "ParamId": 1,
            "DimId": 1,
            "Name": "dim1",
            "TypeId": 101
          },
          "DescrNote": {
            "LangCode": "",
            "Descr": "",
            "Note": ""
          }
        }
      ]
    },
    {
      "Param": {
        "ModelId": 1,
        "ParamId": 2,
        "ParamHid": 46,
        "Name": "StartingSeed",
        "Digest": "_20128171604590133",
        "Rank": 0,
        "TypeId": 4,
        "IsExtendable": false,
        "IsHidden": false,
        "NumCumulated": 0,
        "DbRunTable": "StartingSeed_p_2012819",
        "DbSetTable": "StartingSeed_w_2012819",
        "ImportDigest": "_i0128171604590133"
      },
      "DescrNote": {
        "LangCode": "FR",
        "Descr": "Starting Seed",
        "Note": "Random numbers generator starting seed value"
      },
      "ParamDimsTxt": []
    },
    {
      "Param": {
        "ModelId": 1,
        "ParamId": 3,
        "ParamHid": 47,
        "Name": "salaryFull",
        "Digest": "_20128171604590134",
        "Rank": 1,
        "TypeId": 104,
        "IsExtendable": false,
        "IsHidden": false,
        "NumCumulated": 0,
        "DbRunTable": "salaryFull_p_2012812",
        "DbSetTable": "salaryFull_w_2012812",
        "ImportDigest": "_i0128171604590134"
      },
      "DescrNote": {
        "LangCode": "EN",
        "Descr": "Full or part time by Salary level",
        "Note": ""
      },
      "ParamDimsTxt": [
        {
          "Dim": {
            "ModelId": 1,
            "ParamId": 3,
            "DimId": 0,
            "Name": "dim0",
            "TypeId": 103
```

```
      },
      "DescrNote": {
        "LangCode": "EN",
        "Descr": "Full Dim",
        "Note": ""
      }
    }
  ]
},
{
  "Param": {
    "ModelId": 1,
    "ParamId": 4,
    "ParamHid": 48,
    "Name": "baseSalary",
    "Digest": "_20128171604590135",
    "Rank": 0,
    "TypeId": 104,
    "IsExtendable": false,
    "IsHidden": false,
    "NumCumulated": 0,
    "DbRunTable": "baseSalary_p_2012811",
    "DbSetTable": "baseSalary_w_2012811",
    "ImportDigest": "_i0128171604590135"
  },
  "DescrNote": {
    "LangCode": "EN",
    "Descr": "Base salary level",
    "Note": ""
  },
  "ParamDimsTxt": []
},
{
  "Param": {
    "ModelId": 1,
    "ParamId": 5,
    "ParamHid": 49,
    "Name": "filePath",
    "Digest": "_20128171604590136",
    "Rank": 0,
    "TypeId": 21,
    "IsExtendable": false,
    "IsHidden": false,
    "NumCumulated": 0,
    "DbRunTable": "filePath_p_2012814",
    "DbSetTable": "filePath_w_2012814",
    "ImportDigest": "_i0128171604590136"
  },
  "DescrNote": {
    "LangCode": "EN",
    "Descr": "File path string",
    "Note": ""
  },
  "ParamDimsTxt": []
},
{
  "Param": {
    "ModelId": 1,
    "ParamId": 6,
    "ParamHid": 50,
    "Name": "isOldAge",
    "Digest": "_20128171604590137",
    "Rank": 1,
    "TypeId": 7,
    "IsExtendable": false,
    "IsHidden": false,
    "NumCumulated": 0,
    "DbRunTable": "isOldAge_p_2012815",
    "DbSetTable": "isOldAge_w_2012815",
    "ImportDigest": "_i0128171604590137"
  },
  "DescrNote": {
    "LangCode": "EN",
    "Descr": "Is Old Age",
    "Note": "Is Old Age notes"
  },
  "ParamDimsTxt": [
    {
      "Dim": {
        "ModelId": 1,
        "ParamId": 6,
        "DimId": 0,
        "Name": "dim0",
        "TypeId": 101
      },
      "DescrNote": {
        "LangCode": "EN",
        "Descr": "Age Dim",
```

```
          "Note": "Age Dim notes"
        }
      }
    ]
  }
],
"TableTxt": [
  {
    "Table": {
      "ModelId": 1,
      "TableId": 0,
      "TableHid": 82,
      "Name": "salarySex",
      "Digest": "_20128171604590182",
      "IsUser": false,
      "Rank": 2,
      "IsSparse": true,
      "DbExprTable": "salarySex_v_2012882",
      "DbAccTable": "salarySex_a_2012882",
      "DbAccAllView": "salarySex_d_2012882",
      "ExprPos": 1,
      "IsHidden": false,
      "ImportDigest": "_i0128171604590182"
    },
    "LangCode": "EN",
    "TableDescr": "Salary by Sex",
    "TableNote": "Salary by Sex notes",
    "ExprDescr": "Measure",
    "ExprNote": "Measure notes",
    "TableDimsTxt": [
      {
        "Dim": {
          "ModelId": 1,
          "TableId": 0,
          "DimId": 0,
          "Name": "dim0",
          "TypeId": 103,
          "IsTotal": false,
          "DimSize": 3
        },
        "DescrNote": {
          "LangCode": "EN",
          "Descr": "Salary Dim",
          "Note": "Salary Dim notes"
        }
      },
      {
        "Dim": {
          "ModelId": 1,
          "TableId": 0,
          "DimId": 1,
          "Name": "dim1",
          "TypeId": 102,
          "IsTotal": true,
          "DimSize": 3
        },
        "DescrNote": {
          "LangCode": "EN",
          "Descr": "Sex Dim",
          "Note": "Sex Dim notes"
        }
      }
    ],
    "TableAccTxt": [
      {
        "Acc": {
          "ModelId": 1,
          "TableId": 0,
          "AccId": 0,
          "Name": "acc0",
          "IsDerived": false,
          "SrcAcc": "value_sum()",
          "AccSql": ""
        },
        "DescrNote": {
          "LangCode": "EN",
          "Descr": "Sum of salary by sex",
          "Note": ""
        }
      },
      {
        "Acc": {
          "ModelId": 1,
          "TableId": 0,
          "AccId": 1,
          "Name": "acc1",
          "IsDerived": false,
```

```
      "SrcAcc": "value_count()",
      "AccSql": ""
    },
    "DescrNote": {
      "LangCode": "EN",
      "Descr": "Count of salary by sex",
      "Note": ""
    }
  },
  {
    "Acc": {
      "ModelId": 1,
      "TableId": 0,
      "AccId": 2,
      "Name": "acc2",
      "IsDerived": true,
      "SrcAcc": "acc0 + acc1",
      "AccSql": ""
    },
    "DescrNote": {
      "LangCode": "EN",
      "Descr": "Derived accumulator",
      "Note": ""
    }
  }
],
"TableExprTxt": [
  {
    "Expr": {
      "ModelId": 1,
      "TableId": 0,
      "ExprId": 0,
      "Name": "expr0",
      "Decimals": 4,
      "SrcExpr": "OM_AVG(acc0)",
      "ExprSql": ""
    },
    "DescrNote": {
      "LangCode": "EN",
      "Descr": "Average acc0",
      "Note": "Average on acc0 notes"
    }
  },
  {
    "Expr": {
      "ModelId": 1,
      "TableId": 0,
      "ExprId": 1,
      "Name": "expr1",
      "Decimals": 4,
      "SrcExpr": "OM_SUM(acc1)",
      "ExprSql": ""
    },
    "DescrNote": {
      "LangCode": "EN",
      "Descr": "Sum acc1",
      "Note": ""
    }
  },
  {
    "Expr": {
      "ModelId": 1,
      "TableId": 0,
      "ExprId": 2,
      "Name": "expr2",
      "Decimals": 2,
      "SrcExpr": "OM_MIN(acc0)",
      "ExprSql": ""
    },
    "DescrNote": {
      "LangCode": "EN",
      "Descr": "Min acc0",
      "Note": ""
    }
  },
  {
    "Expr": {
      "ModelId": 1,
      "TableId": 0,
      "ExprId": 3,
      "Name": "expr3",
      "Decimals": 3,
      "SrcExpr": "OM_AVG(acc0 * acc1)",
      "ExprSql": ""
    },
    "DescrNote": {
      "LangCode": "EN",
      "Descr": "Average acc0 * acc1",
```

```
        "Note": ""
      }
    }
  ]
},
{
  "Table": {
    "ModelId": 1,
    "TableId": 1,
    "TableHid": 83,
    "Name": "fullAgeSalary",
    "Digest": "_20128171604590183",
    "IsUser": false,
    "Rank": 3,
    "IsSparse": false,
    "DbExprTable": "fullAgeSalary_v_2012883",
    "DbAccTable": "fullAgeSalary_a_2012883",
    "DbAccAllView": "fullAgeSalary_d_2012883",
    "ExprPos": 1,
    "IsHidden": false,
    "ImportDigest": "_i0128171604590183"
  },
  "LangCode": "EN",
  "TableDescr": "Full Time by Age by Salary Group",
  "TableNote": "Full Time by Age by Salary Group notes",
  "ExprDescr": "Measure",
  "ExprNote": "Measure notes",
  "TableDimsTxt": [
    {
      "Dim": {
        "ModelId": 1,
        "TableId": 1,
        "DimId": 0,
        "Name": "dim0",
        "TypeId": 104,
        "IsTotal": false,
        "DimSize": 2
      },
      "DescrNote": {
        "LangCode": "EN",
        "Descr": "Full Time",
        "Note": "Full or Part Time Dim notes"
      }
    },
    {
      "Dim": {
        "ModelId": 1,
        "TableId": 1,
        "DimId": 1,
        "Name": "dim1",
        "TypeId": 101,
        "IsTotal": true,
        "DimSize": 5
      },
      "DescrNote": {
        "LangCode": "EN",
        "Descr": "Age Dim",
        "Note": "Age Dim notes"
      }
    },
    {
      "Dim": {
        "ModelId": 1,
        "TableId": 1,
        "DimId": 2,
        "Name": "dim2",
        "TypeId": 103,
        "IsTotal": false,
        "DimSize": 3
      },
      "DescrNote": {
        "LangCode": "EN",
        "Descr": "Salary Dim",
        "Note": "Salary Dim notes"
      }
    }
  ],
  "TableAccTxt": [
    {
      "Acc": {
        "ModelId": 1,
        "TableId": 1,
        "AccId": 0,
        "Name": "acc0",
        "IsDerived": false,
        "SrcAcc": "raw_value()",
        "AccSql": ""
      }
```

```
      },
      "DescrNote": {
        "LangCode": "EN",
        "Descr": "Full time salary by age",
        "Note": "Full time salary by age notes"
      }
    }
  ],
  "TableExprTxt": [
    {
      "Expr": {
        "ModelId": 1,
        "TableId": 1,
        "ExprId": 0,
        "Name": "expr0",
        "Decimals": 2,
        "SrcExpr": "OM_AVG(acc0)",
        "ExprSql": ""
      },
      "DescrNote": {
        "LangCode": "EN",
        "Descr": "Average acc0",
        "Note": "Average on acc0 notes"
      }
    }
  ]
},
{
  "Table": {
    "ModelId": 1,
    "TableId": 2,
    "TableHid": 84,
    "Name": "ageSexIncome",
    "Digest": "_20128171604590184",
    "IsUser": false,
    "Rank": 2,
    "IsSparse": false,
    "DbExprTable": "ageSexIncome_v_2012884",
    "DbAccTable": "ageSexIncome_a_2012884",
    "DbAccAllView": "ageSexIncome_d_2012884",
    "ExprPos": 0,
    "IsHidden": false,
    "ImportDigest": "_i0128171604590131"
  },
  "LangCode": "EN",
  "TableDescr": "Age by Sex Income",
  "TableNote": "Age by Sex Income notes",
  "ExprDescr": "Income Measure",
  "ExprNote": "Income Measure notes",
  "TableDimsTxt": [
    {
      "Dim": {
        "ModelId": 1,
        "TableId": 2,
        "DimId": 0,
        "Name": "dim0",
        "TypeId": 101,
        "IsTotal": false,
        "DimSize": 4
      },
      "DescrNote": {
        "LangCode": "EN",
        "Descr": "Age Dim",
        "Note": "Age Dim notes"
      }
    },
    {
      "Dim": {
        "ModelId": 1,
        "TableId": 2,
        "DimId": 1,
        "Name": "dim1",
        "TypeId": 102,
        "IsTotal": false,
        "DimSize": 2
      },
      "DescrNote": {
        "LangCode": "EN",
        "Descr": "Sex Dim",
        "Note": "Sex Dim notes"
      }
    }
  ],
  "TableAccTxt": [
    {
      "Acc": {
        "ModelId": 1,
        "TableId": 2,
```

```
        "AccId": 0,
        "Name": "acc0",
        "IsDerived": false,
        "SrcAcc": "raw_value()",
        "AccSql": ""
      },
      "DescrNote": {
        "LangCode": "EN",
        "Descr": "Income",
        "Note": "Income notes"
      }
    },
    {
      "Acc": {
        "ModelId": 1,
        "TableId": 2,
        "AccId": 1,
        "Name": "acc1",
        "IsDerived": false,
        "SrcAcc": "adjust_value()",
        "AccSql": ""
      },
      "DescrNote": {
        "LangCode": "EN",
        "Descr": "Income adjusted",
        "Note": "Income adjusted notes"
      }
    }
  ],
  "TableExprTxt": [
    {
      "Expr": {
        "ModelId": 1,
        "TableId": 2,
        "ExprId": 0,
        "Name": "expr0",
        "Decimals": 2,
        "SrcExpr": "OM_AVG(acc0)",
        "ExprSql": ""
      },
      "DescrNote": {
        "LangCode": "EN",
        "Descr": "Average acc0",
        "Note": "Average on acc0 notes"
      }
    },
    {
      "Expr": {
        "ModelId": 1,
        "TableId": 2,
        "ExprId": 1,
        "Name": "expr1",
        "Decimals": 3,
        "SrcExpr": "OM_AVG(acc1)",
        "ExprSql": ""
      },
      "DescrNote": {
        "LangCode": "EN",
        "Descr": "Average acc1",
        "Note": "Average on acc1 notes"
      }
    }
  ]
},
{
  "Table": {
    "ModelId": 1,
    "TableId": 3,
    "TableHid": 85,
    "Name": "seedOldAge",
    "Digest": "_20128171604590185",
    "IsUser": false,
    "Rank": 0,
    "IsSparse": false,
    "DbExprTable": "seedOldAge_v_2012885",
    "DbAccTable": "seedOldAge_a_2012885",
    "DbAccAllView": "seedOldAge_d_2012885",
    "ExprPos": 0,
    "IsHidden": false,
    "ImportDigest": "_i0128171604590185"
  },
  "LangCode": "EN",
  "TableDescr": "Seed Old Age",
  "TableNote": "Seed Old Age notes",
  "ExprDescr": "Seed Old Age Measure",
  "ExprNote": "Measure notes",
  "TableDimsTxt": [],
  "TableAccTxt": [
```

```
      {
        "Acc": {
          "ModelId": 1,
          "TableId": 3,
          "AccId": 0,
          "Name": "acc0",
          "IsDerived": false,
          "SrcAcc": "raw_value()",
          "AccSql": ""
        },
        "DescrNote": {
          "LangCode": "EN",
          "Descr": "Seed",
          "Note": "Seed notes"
        }
      }
    ],
    "TableExprTxt": [
      {
        "Expr": {
          "ModelId": 1,
          "TableId": 3,
          "ExprId": 0,
          "Name": "expr0",
          "Decimals": 5,
          "SrcExpr": "OM_AVG(acc0)",
          "ExprSql": ""
        },
        "DescrNote": {
          "LangCode": "EN",
          "Descr": "Average acc0",
          "Note": "Average on acc0 notes"
        }
      }
    ]
  }
],
"GroupTxt": [
  {
    "Group": {
      "ModelId": 1,
      "GroupId": 1,
      "IsParam": true,
      "Name": "AllParameters",
      "IsHidden": false,
      "GroupPc": [
        {
          "ModelId": 1,
          "GroupId": 1,
          "ChildPos": 0,
          "ChildGroupId": 2,
          "ChildLeafId": -1
        },
        {
          "ModelId": 1,
          "GroupId": 1,
          "ChildPos": 1,
          "ChildGroupId": 3,
          "ChildLeafId": -1
        },
        {
          "ModelId": 1,
          "GroupId": 1,
          "ChildPos": 2,
          "ChildGroupId": -1,
          "ChildLeafId": 2
        },
        {
          "ModelId": 1,
          "GroupId": 1,
          "ChildPos": 3,
          "ChildGroupId": -1,
          "ChildLeafId": 5
        }
      ]
    },
    "DescrNote": {
      "LangCode": "EN",
      "Descr": "All parameters",
      "Note": "All model parameters group"
    }
  },
  {
    "Group": {
      "ModelId": 1,
      "GroupId": 2,
      "IsParam": true,
```

```
      "Name": "AgeSexParameters",
      "IsHidden": false,
      "GroupPc": [
        {
          "ModelId": 1,
          "GroupId": 2,
          "ChildPos": 0,
          "ChildGroupId": -1,
          "ChildLeafId": 0
        },
        {
          "ModelId": 1,
          "GroupId": 2,
          "ChildPos": 1,
          "ChildGroupId": -1,
          "ChildLeafId": 1
        },
        {
          "ModelId": 1,
          "GroupId": 2,
          "ChildPos": 2,
          "ChildGroupId": -1,
          "ChildLeafId": 6
        }
      ]
    },
    "DescrNote": {
      "LangCode": "EN",
      "Descr": "Age and Sex parameters",
      "Note": "Age and Sex model parameters group"
    }
  },
  {
    "Group": {
      "ModelId": 1,
      "GroupId": 3,
      "IsParam": true,
      "Name": "SalaryParameters",
      "IsHidden": false,
      "GroupPc": [
        {
          "ModelId": 1,
          "GroupId": 3,
          "ChildPos": 0,
          "ChildGroupId": -1,
          "ChildLeafId": 1
        },
        {
          "ModelId": 1,
          "GroupId": 3,
          "ChildPos": 1,
          "ChildGroupId": -1,
          "ChildLeafId": 3
        },
        {
          "ModelId": 1,
          "GroupId": 3,
          "ChildPos": 2,
          "ChildGroupId": -1,
          "ChildLeafId": 4
        }
      ]
    },
    "DescrNote": {
      "LangCode": "EN",
      "Descr": "Salary parameters",
      "Note": "Salary model parameters group"
    }
  },
  {
    "Group": {
      "ModelId": 1,
      "GroupId": 10,
      "IsParam": false,
      "Name": "AdditionalTables",
      "IsHidden": false,
      "GroupPc": [
        {
          "ModelId": 1,
          "GroupId": 10,
          "ChildPos": 0,
          "ChildGroupId": -1,
          "ChildLeafId": 1
        },
        {
          "ModelId": 1,
          "GroupId": 10,
          "ChildPos": 1,
```

```
          "ChildGroupId": -1,
          "ChildLeafId": 2
        },
        {
          "ModelId": 1,
          "GroupId": 10,
          "ChildPos": 2,
          "ChildGroupId": -1,
          "ChildLeafId": 3
        }
      ]
    },
    "DescrNote": {
      "LangCode": "EN",
      "Descr": "Additional output tables",
      "Note": "Additional output tables group notes"
    }
  }
 ]
}
```

GET model metadata including text in all languages

Get model metadata including text (description and notes) in all languages.

Methods:

```
GET /api/model/:model/text/all
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

Call examples:

```
http://localhost:4040/api/model/modelOne/text/all
http://localhost:4040/api/model/_201208171604590148_/text/all
```

Return example:


```
{
 "Model": {
  "ModelId": 1,
  "Name": "modelOne",
  "Digest": "_201208171604590148_",
  "Type": 0,
  "Version": "1.0",
  "CreateDateTime": "2012-08-17 16:04:59.148",
  "DefaultLangCode": "EN"
 },
 "Type": [
  {
   "ModelId": 1,
   "TypeId": 4,
   "TypeHid": 4,
   "Name": "int",
   "Digest": "_int_",
   "DicId": 0,
   "TotalEnumId": 1,
   "Enum": null
  },
  {
   "ModelId": 1,
   "TypeId": 7,
   "TypeHid": 7,
   "Name": "bool",
   "Digest": "_bool_",
   "DicId": 1,
   "TotalEnumId": 2,
   "Enum": [
    {
     "ModelId": 1,
     "TypeId": 7,
     "EnumId": 0,
     "Name": "false"
    },
    {
     "ModelId": 1,
     "TypeId": 7,
     "EnumId": 1,
     "Name": "true"
    }
   ]
  },
  {
   "ModelId": 1,
   "TypeId": 14,
   "TypeHid": 14,
   "Name": "double",
   "Digest": "_double_",
   "DicId": 0,
   "TotalEnumId": 1,
   "Enum": null
  },
  {
   "ModelId": 1,
```

```
      "ModelId": 1,
      "TypeId": 21,
      "TypeHid": 21,
      "Name": "file",
      "Digest": "_file_",
      "DicId": 0,
      "TotalEnumId": 1,
      "Enum": null
    },
    {
      "ModelId": 1,
      "TypeId": 101,
      "TypeHid": 96,
      "Name": "age",
      "Digest": "_20128171604590121",
      "DicId": 2,
      "TotalEnumId": 500,
      "Enum": [
        {
          "ModelId": 1,
          "TypeId": 101,
          "EnumId": 10,
          "Name": "10-20"
        },
        {
          "ModelId": 1,
          "TypeId": 101,
          "EnumId": 20,
          "Name": "20-30"
        },
        {
          "ModelId": 1,
          "TypeId": 101,
          "EnumId": 30,
          "Name": "30-40"
        },
        {
          "ModelId": 1,
          "TypeId": 101,
          "EnumId": 40,
          "Name": "40+"
        }
      ]
    },
    {
      "ModelId": 1,
      "TypeId": 102,
      "TypeHid": 97,
      "Name": "sex",
      "Digest": "_20128171604590122",
      "DicId": 2,
      "TotalEnumId": 800,
      "Enum": [
        {
          "ModelId": 1,
          "TypeId": 102,
          "EnumId": 0,
          "Name": "M"
        },
        {
          "ModelId": 1,
          "TypeId": 102,
          "EnumId": 1,
          "Name": "F"
        }
      ]
    },
    {
      "ModelId": 1,
      "TypeId": 103,
      "TypeHid": 98,
      "Name": "salary",
      "Digest": "_20128171604590123",
      "DicId": 2,
      "TotalEnumId": 400,
      "Enum": [
        {
          "ModelId": 1,
          "TypeId": 103,
          "EnumId": 100,
          "Name": "L"
        },
        {
          "ModelId": 1,
          "TypeId": 103,
          "EnumId": 200,
          "Name": "M"
        },
        {
```

```
        {
          "ModelId": 1,
          "TypeId": 103,
          "EnumId": 300,
          "Name": "H"
        }
      ]
    },
    {
      "ModelId": 1,
      "TypeId": 104,
      "TypeHid": 99,
      "Name": "full",
      "Digest": "_20128171604590124",
      "DicId": 2,
      "TotalEnumId": 44,
      "Enum": [
        {
          "ModelId": 1,
          "TypeId": 104,
          "EnumId": 22,
          "Name": "Full"
        },
        {
          "ModelId": 1,
          "TypeId": 104,
          "EnumId": 33,
          "Name": "Part"
        }
      ]
    }
  ],
  "Param": [
    {
      "ModelId": 1,
      "ParamId": 0,
      "ParamHid": 44,
      "Name": "ageSex",
      "Digest": "_20128171604590131",
      "Rank": 2,
      "TypeId": 14,
      "IsExtendable": true,
      "IsHidden": false,
      "NumCumulated": 0,
      "DbRunTable": "ageSex_p_2012817",
      "DbSetTable": "ageSex_w_2012817",
      "ImportDigest": "_i0128171604590131",
      "Dim": [
        {
          "ModelId": 1,
          "ParamId": 0,
          "DimId": 0,
          "Name": "dim0",
          "TypeId": 101
        },
        {
          "ModelId": 1,
          "ParamId": 0,
          "DimId": 1,
          "Name": "dim1",
          "TypeId": 102
        }
      ],
      "Import": [
        {
          "ModelId": 1,
          "ParamId": 0,
          "FromName": "ageSexIncome",
          "FromModel": "modelOne",
          "IsSampleDim": false
        }
      ]
    },
    {
      "ModelId": 1,
      "ParamId": 1,
      "ParamHid": 45,
      "Name": "salaryAge",
      "Digest": "_20128171604590132",
      "Rank": 2,
      "TypeId": 4,
      "IsExtendable": false,
      "IsHidden": false,
      "NumCumulated": 0,
      "DbRunTable": "salaryAge_p_2012818",
      "DbSetTable": "salaryAge_w_2012818",
      "ImportDigest": "_i0128171604590132",
      "Dim": [
```

```
    {
      "ModelId": 1,
      "ParamId": 1,
      "DimId": 0,
      "Name": "dim0",
      "TypeId": 103
    },
    {
      "ModelId": 1,
      "ParamId": 1,
      "DimId": 1,
      "Name": "dim1",
      "TypeId": 101
    }
  ],
  "Import": [
    {
      "ModelId": 1,
      "ParamId": 1,
      "FromName": "salaryAge",
      "FromModel": "modelOne",
      "IsSampleDim": false
    }
  ]
},
{
  "ModelId": 1,
  "ParamId": 2,
  "ParamHid": 46,
  "Name": "StartingSeed",
  "Digest": "_20128171604590133",
  "Rank": 0,
  "TypeId": 4,
  "IsExtendable": false,
  "IsHidden": false,
  "NumCumulated": 0,
  "DbRunTable": "StartingSeed_p_2012819",
  "DbSetTable": "StartingSeed_w_2012819",
  "ImportDigest": "_i0128171604590133",
  "Dim": null,
  "Import": [
    {
      "ModelId": 1,
      "ParamId": 2,
      "FromName": "StartingSeed",
      "FromModel": "modelOne",
      "IsSampleDim": false
    }
  ]
},
{
  "ModelId": 1,
  "ParamId": 3,
  "ParamHid": 47,
  "Name": "salaryFull",
  "Digest": "_20128171604590134",
  "Rank": 1,
  "TypeId": 104,
  "IsExtendable": false,
  "IsHidden": false,
  "NumCumulated": 0,
  "DbRunTable": "salaryFull_p_2012812",
  "DbSetTable": "salaryFull_w_2012812",
  "ImportDigest": "_i0128171604590134",
  "Dim": [
    {
      "ModelId": 1,
      "ParamId": 3,
      "DimId": 0,
      "Name": "dim0",
      "TypeId": 103
    }
  ],
  "Import": null
},
{
  "ModelId": 1,
  "ParamId": 4,
  "ParamHid": 48,
  "Name": "baseSalary",
  "Digest": "_20128171604590135",
  "Rank": 0,
  "TypeId": 104,
  "IsExtendable": false,
  "IsHidden": false,
  "NumCumulated": 0,
  "DbRunTable": "baseSalary_p_2012811",
  "DbSetTable": "baseSalary_w_2012811",
```

```
      "DbSetTable": "baseSalary_w_2012811",
      "ImportDigest": "_i0128171604590135",
      "Dim": null,
      "Import": null
    },
    {
      "ModelId": 1,
      "ParamId": 5,
      "ParamHid": 49,
      "Name": "filePath",
      "Digest": "_20128171604590136",
      "Rank": 0,
      "TypeId": 21,
      "IsExtendable": false,
      "IsHidden": false,
      "NumCumulated": 0,
      "DbRunTable": "filePath_p_2012814",
      "DbSetTable": "filePath_w_2012814",
      "ImportDigest": "_i0128171604590136",
      "Dim": null,
      "Import": null
    },
    {
      "ModelId": 1,
      "ParamId": 6,
      "ParamHid": 50,
      "Name": "isOldAge",
      "Digest": "_20128171604590137",
      "Rank": 1,
      "TypeId": 7,
      "IsExtendable": false,
      "IsHidden": false,
      "NumCumulated": 0,
      "DbRunTable": "isOldAge_p_2012815",
      "DbSetTable": "isOldAge_w_2012815",
      "ImportDigest": "_i0128171604590137",
      "Dim": [
        {
          "ModelId": 1,
          "ParamId": 6,
          "DimId": 0,
          "Name": "dim0",
          "TypeId": 101
        }
      ],
      "Import": null
    }
  ],
  "Table": [
    {
      "ModelId": 1,
      "TableId": 0,
      "TableHid": 82,
      "Name": "salarySex",
      "Digest": "_20128171604590182",
      "IsUser": false,
      "Rank": 2,
      "IsSparse": true,
      "DbExprTable": "salarySex_v_2012882",
      "DbAccTable": "salarySex_a_2012882",
      "DbAccAllView": "salarySex_d_2012882",
      "ExprPos": 1,
      "IsHidden": false,
      "ImportDigest": "_i0128171604590182",
      "Dim": [
        {
          "ModelId": 1,
          "TableId": 0,
          "DimId": 0,
          "Name": "dim0",
          "TypeId": 103,
          "IsTotal": false,
          "DimSize": 3
        },
        {
          "ModelId": 1,
          "TableId": 0,
          "DimId": 1,
          "Name": "dim1",
          "TypeId": 102,
          "IsTotal": true,
          "DimSize": 3
        }
      ],
      "Acc": [
        {
          "ModelId": 1,
          "TableId": 0,
```

"AccId": 0,
"Name": "acc0",
"IsDerived": false,
"SrcAcc": "value_sum()",
"AccSql": "A.acc_value"
},
{
"ModelId": 1,
"TableId": 0,
"AccId": 1,
"Name": "acc1",
"IsDerived": false,
"SrcAcc": "value_count()",
"AccSql": "SELECT A1.acc_value FROM salarySex_a_2012882 A1 WHERE A1.run_id = A.run_id AND A1.sub_id = A.sub_id AND A1.dim0 = A.dim0 AND A1.dim1 = A.dim1 AND A1.acc_id = 1"
},
{
"ModelId": 1,
"TableId": 0,
"AccId": 2,
"Name": "acc2",
"IsDerived": true,
"SrcAcc": "acc0 + acc1",
"AccSql": "(A.acc_value) + (SELECT A1.acc_value FROM salarySex_a_2012882 A1 WHERE A1.run_id = A.run_id AND A1.sub_id = A.sub_id AND A1.dim0 = A.dim0 AND A1.dim1 = A.dim1 AND A1.acc_id = 1)"
}
],
"Expr": [
{
"ModelId": 1,
"TableId": 0,
"ExprId": 0,
"Name": "expr0",
"Decimals": 4,
"SrcExpr": "OM_AVG(acc0)",
"ExprSql": "SELECT M1.run_id, M1.dim0, M1.dim1, AVG(M1.acc_value) AS expr0 FROM salarySex_a_2012882 M1 WHERE M1.acc_id = 0 GROUP BY M1.run_id, M1.dim0, M1.dim1"
},
{
"ModelId": 1,
"TableId": 0,
"ExprId": 1,
"Name": "expr1",
"Decimals": 4,
"SrcExpr": "OM_SUM(acc1)",
"ExprSql": "SELECT M1.run_id, M1.dim0, M1.dim1, SUM(M1.acc_value) AS expr1 FROM salarySex_a_2012882 M1 WHERE M1.acc_id = 1 GROUP BY M1.run_id, M1.dim0, M1.dim1"
},
{
"ModelId": 1,
"TableId": 0,
"ExprId": 2,
"Name": "expr2",
"Decimals": 2,
"SrcExpr": "OM_MIN(acc0)",
"ExprSql": "SELECT M1.run_id, M1.dim0, M1.dim1, MIN(M1.acc_value) AS expr2 FROM salarySex_a_2012882 M1 WHERE M1.acc_id = 0 GROUP BY M1.run_id, M1.dim0, M1.dim1"
},
{
"ModelId": 1,
"TableId": 0,
"ExprId": 3,
"Name": "expr3",
"Decimals": 3,
"SrcExpr": "OM_AVG(acc0 * acc1)",
"ExprSql": "SELECT M1.run_id, M1.dim0, M1.dim1, AVG(M1.acc_value * A1.acc1) AS expr3 FROM salarySex_a_2012882 M1 INNER JOIN (SELECT run_id, dim0, dim1, sub_id, acc_value AS acc1 FROM salarySex_a_2012882 WHERE acc_id = 1) A1 ON (A1.run_id = M1.run_id AND A1.dim0 = M1.dim0 AND A1.dim1 = M1.dim1 AND A1.sub_id = M1.sub_id) WHERE M1.acc_id = 0 GROUP BY M1.run_id, M1.dim0, M1.dim1"
}
]
},
{
"ModelId": 1,
"TableId": 1,
"TableHid": 83,
"Name": "fullAgeSalary",
"Digest": "_20128171604590183",
"IsUser": false,
"Rank": 3,
"IsSparse": false,
"DbExprTable": "fullAgeSalary_v_2012883",
"DbAccTable": "fullAgeSalary_a_2012883",
"DbAccAllView": "fullAgeSalary_d_2012883",
"ExprPos": 1,
"IsHidden": false,
"ImportDigest": "_i0128171604590183",
"Dim": [
{

```
    {
     "ModelId": 1,
     "TableId": 1,
     "DimId": 0,
     "Name": "dim0",
     "TypeId": 104,
     "IsTotal": false,
     "DimSize": 2
    },
    {
     "ModelId": 1,
     "TableId": 1,
     "DimId": 1,
     "Name": "dim1",
     "TypeId": 101,
     "IsTotal": true,
     "DimSize": 5
    },
    {
     "ModelId": 1,
     "TableId": 1,
     "DimId": 2,
     "Name": "dim2",
     "TypeId": 103,
     "IsTotal": false,
     "DimSize": 3
    }
   ],
   "Acc": [
    {
     "ModelId": 1,
     "TableId": 1,
     "AccId": 0,
     "Name": "acc0",
     "IsDerived": false,
     "SrcAcc": "raw_value()",
     "AccSql": "A.acc_value"
    }
   ],
   "Expr": [
    {
     "ModelId": 1,
     "TableId": 1,
     "ExprId": 0,
     "Name": "expr0",
     "Decimals": 2,
     "SrcExpr": "OM_AVG(acc0)",
     "ExprSql": "SELECT M1.run_id, M1.dim0, M1.dim1, M1.dim2, AVG(M1.acc_value) AS expr0 FROM fullAgeSalary_a_2012883 M1 WHERE M1.acc_id = 0 GROUP BY M1.run_id, M1.dim0, M1.dim1, M1.dim2"
    }
   ]
  },
  {
   "ModelId": 1,
   "TableId": 2,
   "TableHid": 84,
   "Name": "ageSexIncome",
   "Digest": "_20128171604590184",
   "IsUser": false,
   "Rank": 2,
   "IsSparse": false,
   "DbExprTable": "ageSexIncome_v_2012884",
   "DbAccTable": "ageSexIncome_a_2012884",
   "DbAccAllView": "ageSexIncome_d_2012884",
   "ExprPos": 0,
   "IsHidden": false,
   "ImportDigest": "_i0128171604590131",
   "Dim": [
    {
     "ModelId": 1,
     "TableId": 2,
     "DimId": 0,
     "Name": "dim0",
     "TypeId": 101,
     "IsTotal": false,
     "DimSize": 4
    },
    {
     "ModelId": 1,
     "TableId": 2,
     "DimId": 1,
     "Name": "dim1",
     "TypeId": 102,
     "IsTotal": false,
     "DimSize": 2
    }
   ],
   "Acc": [
    {
```

```
        {
          "ModelId": 1,
          "TableId": 2,
          "AccId": 0,
          "Name": "acc0",
          "IsDerived": false,
          "SrcAcc": "raw_value()",
          "AccSql": "A.acc_value"
        },
        {
          "ModelId": 1,
          "TableId": 2,
          "AccId": 1,
          "Name": "acc1",
          "IsDerived": false,
          "SrcAcc": "adjust_value()",
          "AccSql": "A.acc_value"
        }
      ],
      "Expr": [
        {
          "ModelId": 1,
          "TableId": 2,
          "ExprId": 0,
          "Name": "expr0",
          "Decimals": 2,
          "SrcExpr": "OM_AVG(acc0)",
          "ExprSql": "SELECT M1.run_id, M1.dim0, M1.dim1, AVG(M1.acc_value) AS expr0 FROM ageSexIncome_a_2012884 M1 WHERE M1.acc_id = 0 GROUP BY M1.run_id, M1.d
im0, M1.dim1"
        },
        {
          "ModelId": 1,
          "TableId": 2,
          "ExprId": 1,
          "Name": "expr1",
          "Decimals": 3,
          "SrcExpr": "OM_AVG(acc1)",
          "ExprSql": "SELECT M1.run_id, M1.dim0, M1.dim1, AVG(M1.acc_value) AS expr1 FROM ageSexIncome_a_2012884 M1 WHERE M1.acc_id = 1 GROUP BY M1.run_id, M1.d
im0, M1.dim1"
        }
      ]
    },
    {
      "ModelId": 1,
      "TableId": 3,
      "TableHid": 85,
      "Name": "seedOldAge",
      "Digest": "_20128171604590185",
      "IsUser": false,
      "Rank": 0,
      "IsSparse": false,
      "DbExprTable": "seedOldAge_v_2012885",
      "DbAccTable": "seedOldAge_a_2012885",
      "DbAccAllView": "seedOldAge_d_2012885",
      "ExprPos": 0,
      "IsHidden": false,
      "ImportDigest": "_i0128171604590185",
      "Dim": null,
      "Acc": [
        {
          "ModelId": 1,
          "TableId": 3,
          "AccId": 0,
          "Name": "acc0",
          "IsDerived": false,
          "SrcAcc": "raw_value()",
          "AccSql": "A.acc_value"
        }
      ],
      "Expr": [
        {
          "ModelId": 1,
          "TableId": 3,
          "ExprId": 0,
          "Name": "expr0",
          "Decimals": 5,
          "SrcExpr": "OM_AVG(acc0)",
          "ExprSql": "SELECT M1.run_id, AVG(M1.acc_value) AS expr0 FROM seedOldAge_a_2012885 M1 WHERE M1.acc_id = 0 GROUP BY M1.run_id"
        }
      ]
    }
  ],
  "Group": [
    {
      "ModelId": 1,
      "GroupId": 1,
      "IsParam": true,
      "Name": "AllParameters",
```

```
    "IsHidden": false,
    "GroupPc": [
      {
        "ModelId": 1,
        "GroupId": 1,
        "ChildPos": 0,
        "ChildGroupId": 2,
        "ChildLeafId": -1
      },
      {
        "ModelId": 1,
        "GroupId": 1,
        "ChildPos": 1,
        "ChildGroupId": 3,
        "ChildLeafId": -1
      },
      {
        "ModelId": 1,
        "GroupId": 1,
        "ChildPos": 2,
        "ChildGroupId": -1,
        "ChildLeafId": 2
      },
      {
        "ModelId": 1,
        "GroupId": 1,
        "ChildPos": 3,
        "ChildGroupId": -1,
        "ChildLeafId": 5
      }
    ]
  },
  {
    "ModelId": 1,
    "GroupId": 2,
    "IsParam": true,
    "Name": "AgeSexParameters",
    "IsHidden": false,
    "GroupPc": [
      {
        "ModelId": 1,
        "GroupId": 2,
        "ChildPos": 0,
        "ChildGroupId": -1,
        "ChildLeafId": 0
      },
      {
        "ModelId": 1,
        "GroupId": 2,
        "ChildPos": 1,
        "ChildGroupId": -1,
        "ChildLeafId": 1
      },
      {
        "ModelId": 1,
        "GroupId": 2,
        "ChildPos": 2,
        "ChildGroupId": -1,
        "ChildLeafId": 6
      }
    ]
  },
  {
    "ModelId": 1,
    "GroupId": 3,
    "IsParam": true,
    "Name": "SalaryParameters",
    "IsHidden": false,
    "GroupPc": [
      {
        "ModelId": 1,
        "GroupId": 3,
        "ChildPos": 0,
        "ChildGroupId": -1,
        "ChildLeafId": 1
      },
      {
        "ModelId": 1,
        "GroupId": 3,
        "ChildPos": 1,
        "ChildGroupId": -1,
        "ChildLeafId": 3
      },
      {
        "ModelId": 1,
        "GroupId": 3,
        "ChildPos": 2,
        "ChildGroupId": -1,
```

```
        "ChildGroupId": -1,
        "ChildLeafId": 4
      }
    ]
  },
  {
    "ModelId": 1,
    "GroupId": 10,
    "IsParam": false,
    "Name": "AdditionalTables",
    "IsHidden": false,
    "GroupPc": [
      {
        "ModelId": 1,
        "GroupId": 10,
        "ChildPos": 0,
        "ChildGroupId": -1,
        "ChildLeafId": 1
      },
      {
        "ModelId": 1,
        "GroupId": 10,
        "ChildPos": 1,
        "ChildGroupId": -1,
        "ChildLeafId": 2
      },
      {
        "ModelId": 1,
        "GroupId": 10,
        "ChildPos": 2,
        "ChildGroupId": -1,
        "ChildLeafId": 3
      }
    ]
  }
],
"ModelName": "modelOne",
"ModelDigest": "_201208171604590148_",
"ModelTxt": [
  {
    "ModelId": 1,
    "LangCode": "EN",
    "Descr": "First model",
    "Note": "First model: openM++ development test model"
  },
  {
    "ModelId": 1,
    "LangCode": "FR",
    "Descr": "(FR) First model",
    "Note": ""
  }
],
"TypeTxt": [
  {
    "ModelId": 1,
    "TypeId": 7,
    "LangCode": "EN",
    "Descr": "logical type",
    "Note": ""
  },
  {
    "ModelId": 1,
    "TypeId": 7,
    "LangCode": "FR",
    "Descr": "type logique",
    "Note": ""
  },
  {
    "ModelId": 1,
    "TypeId": 101,
    "LangCode": "EN",
    "Descr": "Age",
    "Note": ""
  },
  {
    "ModelId": 1,
    "TypeId": 101,
    "LangCode": "FR",
    "Descr": "(FR) Age",
    "Note": ""
  },
  {
    "ModelId": 1,
    "TypeId": 102,
    "LangCode": "EN",
    "Descr": "Sex",
    "Note": ""
  },
```

```
    {
      "ModelId": 1,
      "TypeId": 103,
      "LangCode": "EN",
      "Descr": "Salary",
      "Note": ""
    },
    {
      "ModelId": 1,
      "TypeId": 104,
      "LangCode": "EN",
      "Descr": "Full or part time",
      "Note": ""
    }
  ],
  "TypeEnumTxt": [
    {
      "ModelId": 1,
      "TypeId": 7,
      "EnumId": 0,
      "LangCode": "EN",
      "Descr": "False",
      "Note": ""
    },
    {
      "ModelId": 1,
      "TypeId": 7,
      "EnumId": 0,
      "LangCode": "FR",
      "Descr": "Faux",
      "Note": ""
    },
    {
      "ModelId": 1,
      "TypeId": 7,
      "EnumId": 1,
      "LangCode": "EN",
      "Descr": "True",
      "Note": ""
    },
    {
      "ModelId": 1,
      "TypeId": 7,
      "EnumId": 1,
      "LangCode": "FR",
      "Descr": "Vrai",
      "Note": ""
    },
    {
      "ModelId": 1,
      "TypeId": 101,
      "EnumId": 10,
      "LangCode": "EN",
      "Descr": "age 10-20",
      "Note": ""
    },
    {
      "ModelId": 1,
      "TypeId": 101,
      "EnumId": 10,
      "LangCode": "FR",
      "Descr": "(FR) age 10-20",
      "Note": ""
    },
    {
      "ModelId": 1,
      "TypeId": 101,
      "EnumId": 20,
      "LangCode": "EN",
      "Descr": "age 20-30",
      "Note": ""
    },
    {
      "ModelId": 1,
      "TypeId": 101,
      "EnumId": 20,
      "LangCode": "FR",
      "Descr": "(FR) age 20-30",
      "Note": ""
    },
    {
      "ModelId": 1,
      "TypeId": 101,
      "EnumId": 30,
      "LangCode": "EN",
      "Descr": "age 30-40",
      "Note": ""
    }
```

```
  },
  {
    "ModelId": 1,
    "TypeId": 101,
    "EnumId": 30,
    "LangCode": "FR",
    "Descr": "(FR) age 30-40",
    "Note": ""
  },
  {
    "ModelId": 1,
    "TypeId": 101,
    "EnumId": 40,
    "LangCode": "EN",
    "Descr": "age 40+",
    "Note": ""
  },
  {
    "ModelId": 1,
    "TypeId": 101,
    "EnumId": 40,
    "LangCode": "FR",
    "Descr": "(FR) age 40+",
    "Note": ""
  },
  {
    "ModelId": 1,
    "TypeId": 102,
    "EnumId": 0,
    "LangCode": "EN",
    "Descr": "Male",
    "Note": ""
  },
  {
    "ModelId": 1,
    "TypeId": 102,
    "EnumId": 1,
    "LangCode": "EN",
    "Descr": "Female",
    "Note": ""
  },
  {
    "ModelId": 1,
    "TypeId": 103,
    "EnumId": 100,
    "LangCode": "EN",
    "Descr": "Low",
    "Note": ""
  },
  {
    "ModelId": 1,
    "TypeId": 103,
    "EnumId": 200,
    "LangCode": "EN",
    "Descr": "Medium",
    "Note": ""
  },
  {
    "ModelId": 1,
    "TypeId": 103,
    "EnumId": 300,
    "LangCode": "EN",
    "Descr": "High",
    "Note": ""
  },
  {
    "ModelId": 1,
    "TypeId": 104,
    "EnumId": 22,
    "LangCode": "EN",
    "Descr": "Full-time",
    "Note": ""
  },
  {
    "ModelId": 1,
    "TypeId": 104,
    "EnumId": 33,
    "LangCode": "EN",
    "Descr": "Part-time",
    "Note": ""
  }
],
"ParamTxt": [
  {
    "ModelId": 1,
    "ParamId": 0,
    "LangCode": "EN",
    "Descr": "Age by Sex",
```

```
    "Note": "Age by Sex note"
  },
  {
    "ModelId": 1,
    "ParamId": 0,
    "LangCode": "FR",
    "Descr": "(FR) Age by Sex",
    "Note": ""
  },
  {
    "ModelId": 1,
    "ParamId": 1,
    "LangCode": "EN",
    "Descr": "Salary by Age",
    "Note": "Salary by Age note"
  },
  {
    "ModelId": 1,
    "ParamId": 1,
    "LangCode": "FR",
    "Descr": "(FR) Salary by Age",
    "Note": "(FR) Salary by Age note"
  },
  {
    "ModelId": 1,
    "ParamId": 2,
    "LangCode": "FR",
    "Descr": "Starting Seed",
    "Note": "Random numbers generator starting seed value"
  },
  {
    "ModelId": 1,
    "ParamId": 3,
    "LangCode": "EN",
    "Descr": "Full or part time by Salary level",
    "Note": ""
  },
  {
    "ModelId": 1,
    "ParamId": 4,
    "LangCode": "EN",
    "Descr": "Base salary level",
    "Note": ""
  },
  {
    "ModelId": 1,
    "ParamId": 5,
    "LangCode": "EN",
    "Descr": "File path string",
    "Note": ""
  },
  {
    "ModelId": 1,
    "ParamId": 6,
    "LangCode": "EN",
    "Descr": "Is Old Age",
    "Note": "Is Old Age notes"
  },
  {
    "ModelId": 1,
    "ParamId": 6,
    "LangCode": "FR",
    "Descr": "(FR) Is Old Age",
    "Note": "(FR) Is Old Age notes"
  }
 ],
 "ParamDimsTxt": [
  {
    "ModelId": 1,
    "ParamId": 0,
    "DimId": 0,
    "LangCode": "EN",
    "Descr": "Age Dim",
    "Note": "Age Dim notes"
  },
  {
    "ModelId": 1,
    "ParamId": 0,
    "DimId": 0,
    "LangCode": "FR",
    "Descr": "(FR) Age Dim",
    "Note": "(FR) Age Dim notes"
  },
  {
    "ModelId": 1,
    "ParamId": 0,
    "DimId": 1,
    "LangCode": "EN",
```

```
      "Descr": "Sex Dim",
      "Note": "Sex Dim notes"
    },
    {
      "ModelId": 1,
      "ParamId": 0,
      "DimId": 1,
      "LangCode": "FR",
      "Descr": "Sex Dim",
      "Note": ""
    },
    {
      "ModelId": 1,
      "ParamId": 3,
      "DimId": 0,
      "LangCode": "EN",
      "Descr": "Full Dim",
      "Note": ""
    },
    {
      "ModelId": 1,
      "ParamId": 6,
      "DimId": 0,
      "LangCode": "EN",
      "Descr": "Age Dim",
      "Note": "Age Dim notes"
    },
    {
      "ModelId": 1,
      "ParamId": 6,
      "DimId": 0,
      "LangCode": "FR",
      "Descr": "(FR) Age Dim",
      "Note": "(FR) Age Dim notes"
    }
  ],
  "TableTxt": [
    {
      "ModelId": 1,
      "TableId": 0,
      "LangCode": "EN",
      "Descr": "Salary by Sex",
      "Note": "Salary by Sex notes",
      "ExprDescr": "Measure",
      "ExprNote": "Measure notes"
    },
    {
      "ModelId": 1,
      "TableId": 0,
      "LangCode": "FR",
      "Descr": "(FR) Salary by Sex",
      "Note": "(FR) Salary by Sex notes",
      "ExprDescr": "(FR) Measure",
      "ExprNote": ""
    },
    {
      "ModelId": 1,
      "TableId": 1,
      "LangCode": "EN",
      "Descr": "Full Time by Age by Salary Group",
      "Note": "Full Time by Age by Salary Group notes",
      "ExprDescr": "Measure",
      "ExprNote": "Measure notes"
    },
    {
      "ModelId": 1,
      "TableId": 1,
      "LangCode": "FR",
      "Descr": "(FR) Full Time by Age by Salary Group",
      "Note": "(FR) Full Time by Age by Salary Group notes",
      "ExprDescr": "(FR) Measure",
      "ExprNote": ""
    },
    {
      "ModelId": 1,
      "TableId": 2,
      "LangCode": "EN",
      "Descr": "Age by Sex Income",
      "Note": "Age by Sex Income notes",
      "ExprDescr": "Income Measure",
      "ExprNote": "Income Measure notes"
    },
    {
      "ModelId": 1,
      "TableId": 2,
      "LangCode": "FR",
      "Descr": "(FR) Age by Sex Income",
      "Note": "(FR) Age by Sex Income notes"
```

```
      "Note": "(FR) Age by Sex Income notes",
      "ExprDescr": "(FR) Income Measure notes",
      "ExprNote": ""
    },
    {
      "ModelId": 1,
      "TableId": 3,
      "LangCode": "EN",
      "Descr": "Seed Old Age",
      "Note": "Seed Old Age notes",
      "ExprDescr": "Seed Old Age Measure",
      "ExprNote": "Measure notes"
    },
    {
      "ModelId": 1,
      "TableId": 3,
      "LangCode": "FR",
      "Descr": "(FR) Seed Old Age",
      "Note": "(FR) Seed Old Age notes",
      "ExprDescr": "(FR) Measure notes",
      "ExprNote": ""
    }
  ],
  "TableDimsTxt": [
    {
      "ModelId": 1,
      "TableId": 0,
      "DimId": 0,
      "LangCode": "EN",
      "Descr": "Salary Dim",
      "Note": "Salary Dim notes"
    },
    {
      "ModelId": 1,
      "TableId": 0,
      "DimId": 0,
      "LangCode": "FR",
      "Descr": "(FR) Salary Dim",
      "Note": "(FR) Salary Dim notes"
    },
    {
      "ModelId": 1,
      "TableId": 0,
      "DimId": 1,
      "LangCode": "EN",
      "Descr": "Sex Dim",
      "Note": "Sex Dim notes"
    },
    {
      "ModelId": 1,
      "TableId": 0,
      "DimId": 1,
      "LangCode": "FR",
      "Descr": "(FR) Sex Dim",
      "Note": ""
    },
    {
      "ModelId": 1,
      "TableId": 1,
      "DimId": 0,
      "LangCode": "EN",
      "Descr": "Full Time",
      "Note": "Full or Part Time Dim notes"
    },
    {
      "ModelId": 1,
      "TableId": 1,
      "DimId": 0,
      "LangCode": "FR",
      "Descr": "(FR) Full Time",
      "Note": "(FR) Full or Part Time Dim notes"
    },
    {
      "ModelId": 1,
      "TableId": 1,
      "DimId": 1,
      "LangCode": "EN",
      "Descr": "Age Dim",
      "Note": "Age Dim notes"
    },
    {
      "ModelId": 1,
      "TableId": 1,
      "DimId": 1,
      "LangCode": "FR",
      "Descr": "(FR) Age Dim",
      "Note": ""
    },
```

```
    {
      "ModelId": 1,
      "TableId": 1,
      "DimId": 2,
      "LangCode": "EN",
      "Descr": "Salary Dim",
      "Note": "Salary Dim notes"
    },
    {
      "ModelId": 1,
      "TableId": 1,
      "DimId": 2,
      "LangCode": "FR",
      "Descr": "(FR) Salary Dim",
      "Note": "(FR) Salary Dim notes"
    },
    {
      "ModelId": 1,
      "TableId": 2,
      "DimId": 0,
      "LangCode": "EN",
      "Descr": "Age Dim",
      "Note": "Age Dim notes"
    },
    {
      "ModelId": 1,
      "TableId": 2,
      "DimId": 0,
      "LangCode": "FR",
      "Descr": "(FR) Age Dim",
      "Note": "(FR) Age Dim notes"
    },
    {
      "ModelId": 1,
      "TableId": 2,
      "DimId": 1,
      "LangCode": "EN",
      "Descr": "Sex Dim",
      "Note": "Sex Dim notes"
    },
    {
      "ModelId": 1,
      "TableId": 2,
      "DimId": 1,
      "LangCode": "FR",
      "Descr": "(FR) Sex Dim",
      "Note": ""
    }
  ],
  "TableAccTxt": [
    {
      "ModelId": 1,
      "TableId": 0,
      "AccId": 0,
      "LangCode": "EN",
      "Descr": "Sum of salary by sex",
      "Note": ""
    },
    {
      "ModelId": 1,
      "TableId": 0,
      "AccId": 1,
      "LangCode": "EN",
      "Descr": "Count of salary by sex",
      "Note": ""
    },
    {
      "ModelId": 1,
      "TableId": 0,
      "AccId": 2,
      "LangCode": "EN",
      "Descr": "Derived accumulator",
      "Note": ""
    },
    {
      "ModelId": 1,
      "TableId": 1,
      "AccId": 0,
      "LangCode": "EN",
      "Descr": "Full time salary by age",
      "Note": "Full time salary by age notes"
    },
    {
      "ModelId": 1,
      "TableId": 2,
      "AccId": 0,
      "LangCode": "EN",
      "Descr": "Income",
```

```
      "Note": "Income notes"
    },
    {
      "ModelId": 1,
      "TableId": 2,
      "AccId": 1,
      "LangCode": "EN",
      "Descr": "Income adjusted",
      "Note": "Income adjusted notes"
    },
    {
      "ModelId": 1,
      "TableId": 3,
      "AccId": 0,
      "LangCode": "EN",
      "Descr": "Seed",
      "Note": "Seed notes"
    }
  ],
  "TableExprTxt": [
    {
      "ModelId": 1,
      "TableId": 0,
      "ExprId": 0,
      "LangCode": "EN",
      "Descr": "Average acc0",
      "Note": "Average on acc0 notes"
    },
    {
      "ModelId": 1,
      "TableId": 0,
      "ExprId": 0,
      "LangCode": "FR",
      "Descr": "(FR) Average acc0",
      "Note": "(FR) Average on acc0 notes"
    },
    {
      "ModelId": 1,
      "TableId": 0,
      "ExprId": 1,
      "LangCode": "EN",
      "Descr": "Sum acc1",
      "Note": ""
    },
    {
      "ModelId": 1,
      "TableId": 0,
      "ExprId": 2,
      "LangCode": "EN",
      "Descr": "Min acc0",
      "Note": ""
    },
    {
      "ModelId": 1,
      "TableId": 0,
      "ExprId": 3,
      "LangCode": "EN",
      "Descr": "Average acc0 * acc1",
      "Note": ""
    },
    {
      "ModelId": 1,
      "TableId": 1,
      "ExprId": 0,
      "LangCode": "EN",
      "Descr": "Average acc0",
      "Note": "Average on acc0 notes"
    },
    {
      "ModelId": 1,
      "TableId": 2,
      "ExprId": 0,
      "LangCode": "EN",
      "Descr": "Average acc0",
      "Note": "Average on acc0 notes"
    },
    {
      "ModelId": 1,
      "TableId": 2,
      "ExprId": 1,
      "LangCode": "EN",
      "Descr": "Average acc1",
      "Note": "Average on acc1 notes"
    },
    {
      "ModelId": 1,
      "TableId": 3,
```

```
      "ExprId": 0,
      "LangCode": "EN",
      "Descr": "Average acc0",
      "Note": "Average on acc0 notes"
    }
  ],
  "GroupTxt": [
    {
      "ModelId": 1,
      "GroupId": 1,
      "LangCode": "EN",
      "Descr": "All parameters",
      "Note": "All model parameters group"
    },
    {
      "ModelId": 1,
      "GroupId": 1,
      "LangCode": "FR",
      "Descr": "(FR) All parameters",
      "Note": ""
    },
    {
      "ModelId": 1,
      "GroupId": 2,
      "LangCode": "EN",
      "Descr": "Age and Sex parameters",
      "Note": "Age and Sex model parameters group"
    },
    {
      "ModelId": 1,
      "GroupId": 2,
      "LangCode": "FR",
      "Descr": "(FR) Age and Sex parameters",
      "Note": "(FR) Age and Sex model parameters group"
    },
    {
      "ModelId": 1,
      "GroupId": 3,
      "LangCode": "EN",
      "Descr": "Salary parameters",
      "Note": "Salary model parameters group"
    },
    {
      "ModelId": 1,
      "GroupId": 10,
      "LangCode": "EN",
      "Descr": "Additional output tables",
      "Note": "Additional output tables group notes"
    },
    {
      "ModelId": 1,
      "GroupId": 10,
      "LangCode": "FR",
      "Descr": "(FR) Additional output tables",
      "Note": ""
    }
  ]
}
```

GET model languages

Get model languages.

Methods:

```
GET /api/model/:model/lang-list
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

Call examples:

```
http://localhost:4040/api/model/modelOne/lang-list
http://localhost:4040/api/model/649f17f26d67c37b78dde94f79772445/lang-list
```

Return example:

Known issue: There is no "model languages" table in current database, only master language list table `lang_lst`. As result if there are multiple model in same database it is assumed all models have same list of languages.

```
[
  {
    "LangCode": "EN",
    "Name": "English"
  },
  {
    "LangCode": "FR",
    "Name": "Français"
  }
]
```

GET model language-specific strings

Get model language-specific strings.

Language-specific strings are (code, label) rows from `lang_word` and `model_word` database tables.

Methods:

```
GET /api/model/:model/word-list
GET /api/model/:model/word-list/lang/:lang
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:lang - (optional) language code
```

If optional lang argument specified then result in that language else in browser language or model default. If no such language exist then result in model default language or can be empty.

Call examples:

```
http://localhost:4040/api/model/modelOne/word-list
http://localhost:4040/api/model/modelOne/word-list/lang/fr-CA
http://localhost:4040/api/model/_201208171604590148_/word-list
```

Return example:

```
{
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "LangCode": "EN",
  "LangWords": [
    {
      "Code": "all",
      "Label": "All"
    },
    {
      "Code": "max",
      "Label": "Max"
    },
    {
      "Code": "min",
      "Label": "Min"
    },
    {
      "Code": "Sub-value %d",
      "Label": "Sub-value %d"
    },
    {
      "Code": "Read",
      "Label": "Read"
    }
  ],
  "ModelLangCode": "EN",
  "ModelWords": [
    {
      "Code": "Event loop completed",
      "Label": "Event loop completed"
    },
    {
      "Code": "Reading Parameters",
      "Label": "Reading Parameters"
    },
    {
      "Code": "Running Simulation",
      "Label": "Running Simulation"
    },
    {
      "Code": "Start model subvalue",
      "Label": "Start model subvalue"
    },
    {
      "Code": "Writing Output Tables",
      "Label": "Writing Output Tables"
    }
  ]
}
```

GET model profile

Get model profile. Profile is a set of key-value options, similar to ini-file, which can be used to run the model. Please keep in mind, there is no actual link between profiles and models and any profile can be applied to run any model (it is by design, similar to ini-file).

Methods:

```
GET /api/model/:model/profile/:profile
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:profile - (required) profile name
```

Profile name is unique per database.

Call examples:

```
http://localhost:4040/api/model/modelOne/profile/modelOne
```

Return example: *This is a beta version and may change in the future.*


```
{
  "Name": "modelOne",
  "Opts": {
    "OpenM.SparseOutput": "true",
    "Parameter.StartingSeed": "1023"
  }
}
```

GET list of profiles

Get list of profile names by model name or model digest.

Profile is a set of key-value options, similar to ini-file, which can be used to run the model. Please keep in mind, there is no actual link between profiles and models and any profile can be applied to run any model (it is by design, similar to ini-file).

Methods:

```
GET /api/model/:model/profile-list
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

Model digest or name is used by server to find SQLite database. There is no explicit link between model and profile. All profile name from that database will be selected.

Call examples:

```
http://localhost:4040/api/model/modelOne/profile-list
```

Return example:

```
[
  "modelOne"
]
```

GET list of model runs

Get list of model run results: language-neutral part of run list metadata.

Methods:

```
GET /api/model/:model/run-list
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

Call examples:

```
http://localhost:4040/api/model/modelOne/run-list
http://localhost:4040/api/model/_201208171604590148_/run-list
```

Return example:


```
[
 {
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "Default",
  "SubCount": 1,
  "SubStarted": 1,
  "SubCompleted": 1,
  "CreateDateTime": "2021-03-11 00:27:56.583",
  "Status": "s",
  "UpdateDateTime": "2021-03-11 00:27:57.030",
  "RunDigest": "88b8c45b77993133b07a7c85e4447d5c",
  "ValueDigest": "6c5c0f48e19f67899c868688bb8a23fd",
  "RunStamp": "2021_03_11_00_27_56_535",
  "Txt": [],
  "Opts": {},
  "Param": [],
  "Table": [],
  "Progress": []
 },
 {
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "Default-4",
  "SubCount": 4,
  "SubStarted": 4,
  "SubCompleted": 4,
  "CreateDateTime": "2021-03-11 00:27:57.119",
  "Status": "s",
  "UpdateDateTime": "2021-03-11 00:27:57.955",
  "RunDigest": "c6ced1efa64dca8a98e5cd323ac7f50d",
  "ValueDigest": "d900353af61f7f824ddae66b47b456ea",
  "RunStamp": "2021_03_11_00_27_57_080",
  "Txt": [],
  "Opts": {},
  "Param": [],
  "Table": [],
  "Progress": []
 },
 {
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "First Task Run_Default",
  "SubCount": 1,
  "SubStarted": 1,
  "SubCompleted": 1,
  "CreateDateTime": "2021-03-11 00:27:58.054",
  "Status": "s",
  "UpdateDateTime": "2021-03-11 00:27:58.485",
  "RunDigest": "419f0d1b7078cff499f87be5d9e8995c",
  "ValueDigest": "6c5c0f48e19f67899c868688bb8a23fd",
  "RunStamp": "2021_03_11_00_27_58_005",
  "Txt": [],
  "Opts": {},
  "Param": [],
  "Table": [],
```

```
    "Progress": []
  },
  {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "First Task Run_modelOne_other",
    "SubCount": 1,
    "SubStarted": 1,
    "SubCompleted": 1,
    "CreateDateTime": "2021-03-11 00:27:58.505",
    "Status": "s",
    "UpdateDateTime": "2021-03-11 00:27:58.833",
    "RunDigest": "ec2455261ede37787150692c460a2688",
    "ValueDigest": "fb27d108fae2040fa1cae6f49704a1b7",
    "RunStamp": "2021_03_11_00_27_58_005",
    "Txt": [],
    "Opts": {},
    "Param": [],
    "Table": [],
    "Progress": []
  },
  {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "Sub-values_2_from_csv",
    "SubCount": 2,
    "SubStarted": 2,
    "SubCompleted": 2,
    "CreateDateTime": "2021-03-11 00:27:58.935",
    "Status": "s",
    "UpdateDateTime": "2021-03-11 00:27:59.531",
    "RunDigest": "de486efd4c8d002036876a3b9a285f63",
    "ValueDigest": "c91cee4876452c95717b8d2d6aaee7a5",
    "RunStamp": "2021_03_11_00_27_58_895",
    "Txt": [],
    "Opts": {},
    "Param": [],
    "Table": [],
    "Progress": []
  },
  {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "Sub-values_4",
    "SubCount": 4,
    "SubStarted": 4,
    "SubCompleted": 4,
    "CreateDateTime": "2021-03-11 00:27:59.631",
    "Status": "s",
    "UpdateDateTime": "2021-03-11 00:28:00.492",
    "RunDigest": "668da5c876e3c7c8742d24e17071505f",
    "ValueDigest": "2ccb8ebabceb2cfb23bbca6403ac52d0",
    "RunStamp": "2021_03_11_00_27_59_582",
    "Txt": [],
    "Opts": {},
    "Param": [],
    "Table": [],
    "Progress": []
  },
  {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "Group_sub-values_2_from_csv",
    "SubCount": 2,
    "SubStarted": 2,
    "SubCompleted": 2,
    "CreateDateTime": "2021-03-11 00:28:00.587",
    "Status": "s",
    "UpdateDateTime": "2021-03-11 00:28:00.921",
    "RunDigest": "e36f2fbff9439a8f4f7268e50eef2986",
    "ValueDigest": "d73a023253e620a3df7fc45b4b826a60",
    "RunStamp": "2021_03_11_00_28_00_543",
    "Txt": [],
    "Opts": {},
    "Param": [],
    "Table": [],
    "Progress": []
  },
  {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "Import_from_Default_run",
    "SubCount": 1,
    "SubStarted": 1,
    "SubCompleted": 1,
    "CreateDateTime": "2021-03-11 00:28:01.015",
    "Status": "s",
```

"UpdateDateTime": "2021-03-11 00:28:01.256",
"RunDigest": "dcc2a68b7e86267d7efad9f8b7fd2092",
"ValueDigest": "6c5c0f48e19f67899c868688bb8a23fd",
"RunStamp": "2021_03_11_00_28_00_952",
"Txt": [],
"Opts": {},
"Param": [],
"Table": [],
"Progress": []
},
{
"ModelName": "modelOne",
"ModelDigest": "_201208171604590148_",
"Name": "Base_run_is_Sub-values_2_from_csv",
"SubCount": 2,
"SubStarted": 2,
"SubCompleted": 2,
"CreateDateTime": "2021-03-11 00:28:01.326",
"Status": "s",
"UpdateDateTime": "2021-03-11 00:28:01.619",
"RunDigest": "a57ac3d4c0cefdc09939ad7150661bed",
"ValueDigest": "c91cee4876452c95717b8d2d6aaee7a5",
"RunStamp": "2021_03_11_00_28_01_286",
"Txt": [],
"Opts": {},
"Param": [],
"Table": [],
"Progress": []
},
{
"ModelName": "modelOne",
"ModelDigest": "_201208171604590148_",
"Name": "Base_run_and_partial_input_set",
"SubCount": 1,
"SubStarted": 1,
"SubCompleted": 1,
"CreateDateTime": "2021-03-11 00:28:01.704",
"Status": "s",
"UpdateDateTime": "2021-03-11 00:28:01.913",
"RunDigest": "f170ec1ad8596d1f82114285c3d93eec",
"ValueDigest": "f8638fcc86441f3fd22b2c37e0ed5e47",
"RunStamp": "2021_03_11_00_28_01_661",
"Txt": [],
"Opts": {},
"Param": [],
"Table": [],
"Progress": []
},
{
"ModelName": "modelOne",
"ModelDigest": "_201208171604590148_",
"Name": "Task Run with Suppressed Tables_Default",
"SubCount": 2,
"SubStarted": 2,
"SubCompleted": 2,
"CreateDateTime": "2021-03-11 00:28:01.994",
"Status": "s",
"UpdateDateTime": "2021-03-11 00:28:02.241",
"RunDigest": "e40a172f046a248d85f0fc600d9aa133",
"ValueDigest": "74dc31c98dd0e491bfdbf0f68961576d",
"RunStamp": "2021_03_11_00_28_01_943",
"Txt": [],
"Opts": {},
"Param": [],
"Table": [],
"Progress": []
},
{
"ModelName": "modelOne",
"ModelDigest": "_201208171604590148_",
"Name": "Task Run with Suppressed Tables_modelOne_other",
"SubCount": 2,
"SubStarted": 2,
"SubCompleted": 2,
"CreateDateTime": "2021-03-11 00:28:02.253",
"Status": "s",
"UpdateDateTime": "2021-03-11 00:28:02.435",
"RunDigest": "e97dc09e7ae4965a47688eb90ba434c1",
"ValueDigest": "7dd0761dcfd04cb8def60c63a2804157",
"RunStamp": "2021_03_11_00_28_01_943",
"Txt": [],
"Opts": {},
"Param": [],
"Table": [],
"Progress": []
},
{
"ModelName": "modelOne",

```
    "ModelDigest": "_201208171604590148_",
    "Name": "Task Run with NotSuppressed Tables_Default",
    "SubCount": 2,
    "SubStarted": 2,
    "SubCompleted": 2,
    "CreateDateTime": "2021-03-11 00:28:02.572",
    "Status": "s",
    "UpdateDateTime": "2021-03-11 00:28:03.016",
    "RunDigest": "ef9920516d16859e1705574d7e6f8891",
    "ValueDigest": "e284bb8c7f1e28aa6dc5b52fa78d975d",
    "RunStamp": "2021_03_11_00_28_02_520",
    "Txt": [],
    "Opts": {},
    "Param": [],
    "Table": [],
    "Progress": []
  },
  {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "Task Run with NotSuppressed Tables_modelOne_other",
    "SubCount": 2,
    "SubStarted": 2,
    "SubCompleted": 2,
    "CreateDateTime": "2021-03-11 00:28:03.036",
    "Status": "s",
    "UpdateDateTime": "2021-03-11 00:28:03.372",
    "RunDigest": "a5b56959d3f3efd82e7702289af43022",
    "ValueDigest": "79c55110928e7d372c0570cfa2202867",
    "RunStamp": "2021_03_11_00_28_02_520",
    "Txt": [],
    "Opts": {},
    "Param": [],
    "Table": [],
    "Progress": []
  }
]
```

GET list of model runs including text (description and notes)

Get list of model runs, including text (description and notes).

Methods:

```
GET /api/model/:model/run-list/text
GET /api/model/:model/run-list/text/lang/:lang
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:lang - (optional) language code
```

If optional `lang` argument specified then result in that language else in browser language. If no such language exist then text portion of result (description and notes) is empty.

Call examples:

```
http://localhost:4040/api/model/modelOne/run-list/text
http://localhost:4040/api/model/_201208171604590148_/run-list/text
http://localhost:4040/api/model/modelOne/run-list/text/lang/en_CA
```

Return example:


```
[
 {
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "Default",
  "SubCount": 1,
  "SubStarted": 1,
  "SubCompleted": 1,
  "CreateDateTime": "2021-03-11 00:27:56.583",
  "Status": "s",
  "UpdateDateTime": "2021-03-11 00:27:57.030",
  "RunDigest": "88b8c45b77993133b07a7c85e4447d5c",
  "ValueDigest": "6c5c0f48e19f67899c868688bb8a23fd",
  "RunStamp": "2021_03_11_00_27_56_535",
  "Txt": [
   {
    "LangCode": "EN",
    "Descr": "Model One default set of parameters",
    "Note": ""
   }
  ],
  "Opts": {},
  "Param": [],
  "Table": [],
  "Progress": []
 },
 {
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "Default-4",
  "SubCount": 4,
  "SubStarted": 4,
  "SubCompleted": 4,
  "CreateDateTime": "2021-03-11 00:27:57.119",
  "Status": "s",
  "UpdateDateTime": "2021-03-11 00:27:57.955",
  "RunDigest": "c6ced1efa64dca8a98e5cd323ac7f50d",
  "ValueDigest": "d900353af61f7f824ddae66b47b456ea",
  "RunStamp": "2021_03_11_00_27_57_080",
  "Txt": [
   {
    "LangCode": "EN",
    "Descr": "Model One default set of parameters",
    "Note": ""
   }
  ],
```

```
    "Opts": {},
    "Param": [],
    "Table": [],
    "Progress": []
  },
  {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "First Task Run_Default",
    "SubCount": 1,
    "SubStarted": 1,
    "SubCompleted": 1,
    "CreateDateTime": "2021-03-11 00:27:58.054",
    "Status": "s",
    "UpdateDateTime": "2021-03-11 00:27:58.485",
    "RunDigest": "419f0d1b7078cff499f87be5d9e8995c",
    "ValueDigest": "6c5c0f48e19f67899c868688bb8a23fd",
    "RunStamp": "2021_03_11_00_27_58_005",
    "Txt": [
      {
        "LangCode": "EN",
        "Descr": "Model One default set of parameters",
        "Note": ""
      }
    ],
    "Opts": {},
    "Param": [],
    "Table": [],
    "Progress": []
  },
  {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "First Task Run_modelOne_other",
    "SubCount": 1,
    "SubStarted": 1,
    "SubCompleted": 1,
    "CreateDateTime": "2021-03-11 00:27:58.505",
    "Status": "s",
    "UpdateDateTime": "2021-03-11 00:27:58.833",
    "RunDigest": "ec2455261ede37787150692c460a2688",
    "ValueDigest": "fb27d108fae2040fa1cae6f49704a1b7",
    "RunStamp": "2021_03_11_00_27_58_005",
    "Txt": [
      {
        "LangCode": "EN",
        "Descr": "Model One other set of parameters",
        "Note": ""
      }
    ],
    "Opts": {},
    "Param": [],
    "Table": [],
    "Progress": []
  },
  {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "Sub-values_2_from_csv",
    "SubCount": 2,
    "SubStarted": 2,
    "SubCompleted": 2,
    "CreateDateTime": "2021-03-11 00:27:58.935",
    "Status": "s",
    "UpdateDateTime": "2021-03-11 00:27:59.531",
    "RunDigest": "de486efd4c8d002036876a3b9a285f63",
    "ValueDigest": "c91cee4876452c95717b8d2d6aaee7a5",
    "RunStamp": "2021_03_11_00_27_58_895",
    "Txt": [
      {
        "LangCode": "EN",
        "Descr": "Parameter sub-values 2 from csv",
        "Note": ""
      }
    ],
    "Opts": {},
    "Param": [],
    "Table": [],
    "Progress": []
  },
  {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "Sub-values_4",
    "SubCount": 4,
    "SubStarted": 4,
    "SubCompleted": 4,
    "CreateDateTime": "2021-03-11 00:27:59.631",
```

```
    "Status": "s",
    "UpdateDateTime": "2021-03-11 00:28:00.492",
    "RunDigest": "668da5c876e3c7c8742d24e17071505f",
    "ValueDigest": "2ccb8ebabceb2cfb23bbca6403ac52d0",
    "RunStamp": "2021_03_11_00_27_59_582",
    "Txt": [
      {
        "LangCode": "EN",
        "Descr": "Parameter sub-values 4",
        "Note": ""
      }
    ],
    "Opts": {},
    "Param": [],
    "Table": [],
    "Progress": []
  },
  {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "Group_sub-values_2_from_csv",
    "SubCount": 2,
    "SubStarted": 2,
    "SubCompleted": 2,
    "CreateDateTime": "2021-03-11 00:28:00.587",
    "Status": "s",
    "UpdateDateTime": "2021-03-11 00:28:00.921",
    "RunDigest": "e36f2fbff9439a8f4f7268e50eef2986",
    "ValueDigest": "d73a023253e620a3df7fc45b4b826a60",
    "RunStamp": "2021_03_11_00_28_00_543",
    "Txt": [
      {
        "LangCode": "EN",
        "Descr": "Parameter group sub-values 2 from csv",
        "Note": ""
      }
    ],
    "Opts": {},
    "Param": [],
    "Table": [],
    "Progress": []
  },
  {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "Import_from_Default_run",
    "SubCount": 1,
    "SubStarted": 1,
    "SubCompleted": 1,
    "CreateDateTime": "2021-03-11 00:28:01.015",
    "Status": "s",
    "UpdateDateTime": "2021-03-11 00:28:01.256",
    "RunDigest": "dcc2a68b7e86267d7efad9f8b7fd2092",
    "ValueDigest": "6c5c0f48e19f67899c868688bb8a23fd",
    "RunStamp": "2021_03_11_00_28_00_952",
    "Txt": [
      {
        "LangCode": "EN",
        "Descr": "Import parameters from Default run",
        "Note": ""
      }
    ],
    "Opts": {},
    "Param": [],
    "Table": [],
    "Progress": []
  },
  {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "Base_run_is_Sub-values_2_from_csv",
    "SubCount": 2,
    "SubStarted": 2,
    "SubCompleted": 2,
    "CreateDateTime": "2021-03-11 00:28:01.326",
    "Status": "s",
    "UpdateDateTime": "2021-03-11 00:28:01.619",
    "RunDigest": "a57ac3d4c0cefdc09939ad7150661bed",
    "ValueDigest": "c91cee4876452c95717b8d2d6aaee7a5",
    "RunStamp": "2021_03_11_00_28_01_286",
    "Txt": [
      {
        "LangCode": "EN",
        "Descr": "Parameters from base run Sub-values_2_from_csv",
        "Note": ""
      }
    ],
```

"Opts": {},
"Param": [],
"Table": [],
"Progress": []
},
{
"ModelName": "modelOne",
"ModelDigest": "_201208171604590148_",
"Name": "Base_run_and_partial_input_set",
"SubCount": 1,
"SubStarted": 1,
"SubCompleted": 1,
"CreateDateTime": "2021-03-11 00:28:01.704",
"Status": "s",
"UpdateDateTime": "2021-03-11 00:28:01.913",
"RunDigest": "f170ec1ad8596d1f82114285c3d93eec",
"ValueDigest": "f8638fcc86441f3fd22b2c37e0ed5e47",
"RunStamp": "2021_03_11_00_28_01_661",
"Txt": [
{
"LangCode": "EN",
"Descr": "Parameters from base run and from partial input set",
"Note": ""
}
],
"Opts": {},
"Param": [],
"Table": [],
"Progress": []
},
{
"ModelName": "modelOne",
"ModelDigest": "_201208171604590148_",
"Name": "Task Run with Suppressed Tables_Default",
"SubCount": 2,
"SubStarted": 2,
"SubCompleted": 2,
"CreateDateTime": "2021-03-11 00:28:01.994",
"Status": "s",
"UpdateDateTime": "2021-03-11 00:28:02.241",
"RunDigest": "e40a172f046a248d85f0fc600d9aa133",
"ValueDigest": "74dc31c98dd0e491bfdbf0f68961576d",
"RunStamp": "2021_03_11_00_28_01_943",
"Txt": [
{
"LangCode": "EN",
"Descr": "Model One default set of parameters",
"Note": ""
}
],
"Opts": {},
"Param": [],
"Table": [],
"Progress": []
},
{
"ModelName": "modelOne",
"ModelDigest": "_201208171604590148_",
"Name": "Task Run with Suppressed Tables_modelOne_other",
"SubCount": 2,
"SubStarted": 2,
"SubCompleted": 2,
"CreateDateTime": "2021-03-11 00:28:02.253",
"Status": "s",
"UpdateDateTime": "2021-03-11 00:28:02.435",
"RunDigest": "e97dc09e7ae4965a47688eb90ba434c1",
"ValueDigest": "7dd0761dcfd04cb8def60c63a2804157",
"RunStamp": "2021_03_11_00_28_01_943",
"Txt": [
{
"LangCode": "EN",
"Descr": "Model One other set of parameters",
"Note": ""
}
],
"Opts": {},
"Param": [],
"Table": [],
"Progress": []
},
{
"ModelName": "modelOne",
"ModelDigest": "_201208171604590148_",
"Name": "Task Run with NotSuppressed Tables_Default",
"SubCount": 2,
"SubStarted": 2,
"SubCompleted": 2,
"CreateDateTime": "2021-03-11 00:28:02.572",

```
    "Status": "s",
    "UpdateDateTime": "2021-03-11 00:28:03.016",
    "RunDigest": "ef9920516d16859e1705574d7e6f8891",
    "ValueDigest": "e284bb8c7f1e28aa6dc5b52fa78d975d",
    "RunStamp": "2021_03_11_00_28_02_520",
    "Txt": [
      {
        "LangCode": "EN",
        "Descr": "Model One default set of parameters",
        "Note": ""
      }
    ],
    "Opts": {},
    "Param": [],
    "Table": [],
    "Progress": []
  },
  {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "Task Run with NotSuppressed Tables_modelOne_other",
    "SubCount": 2,
    "SubStarted": 2,
    "SubCompleted": 2,
    "CreateDateTime": "2021-03-11 00:28:03.036",
    "Status": "s",
    "UpdateDateTime": "2021-03-11 00:28:03.372",
    "RunDigest": "a5b56959d3f3efd82e7702289af43022",
    "ValueDigest": "79c55110928e7d372c0570cfa2202867",
    "RunStamp": "2021_03_11_00_28_02_520",
    "Txt": [
      {
        "LangCode": "EN",
        "Descr": "Model One other set of parameters",
        "Note": ""
      }
    ],
    "Opts": {},
    "Param": [],
    "Table": [],
    "Progress": []
  }
]
```

GET status of model run

Get status of model run by run digest, run stamp or run name. If there is only multiple runs with such stamp or name exist then it is better to use GET status of model run list method to get run status of all runs.

Methods:

```
GET /api/model/:model/run/:run/status
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use run name, which is more human readable than digest, but if there are multiple runs with same name or same run stamp in database then result is undefined.

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Default-4/status
http://localhost:4040/api/model/modelOne/run/05403de52f30f59b050417561914fbb8/status
http://localhost:4040/api/model/modelOne/run/2019_01_17_19_59_52_998/status
```

Return example: *This is a beta version and may change in the future.*

```
{
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "Default-4",
  "SubCount": 4,
  "SubStarted": 4,
  "SubCompleted": 4,
  "CreateDateTime": "2021-03-11 00:27:57.119",
  "Status": "s",
  "UpdateDateTime": "2021-03-11 00:27:57.955",
  "RunDigest": "c6ced1efa64dca8a98e5cd323ac7f50d",
  "ValueDigest": "d900353af61f7f824ddae66b47b456ea",
  "RunStamp": "2021_03_11_00_27_57_080",
  "Txt": [],
  "Opts": {},
  "Param": [],
  "Table": [],
  "Progress": [
    {
      "SubId": 0,
      "CreateDateTime": "2021-03-11 00:27:57.151",
      "Status": "s",
      "UpdateDateTime": "2021-03-11 00:27:57.512",
      "Count": 100,
      "Value": 0
    },
    {
      "SubId": 1,
      "CreateDateTime": "2021-03-11 00:27:57.153",
      "Status": "s",
      "UpdateDateTime": "2021-03-11 00:27:57.669",
      "Count": 100,
      "Value": 0
    },
    {
      "SubId": 2,
      "CreateDateTime": "2021-03-11 00:27:57.157",
      "Status": "s",
      "UpdateDateTime": "2021-03-11 00:27:57.649",
      "Count": 100,
      "Value": 0
    },
    {
      "SubId": 3,
      "CreateDateTime": "2021-03-11 00:27:57.159",
      "Status": "s",
      "UpdateDateTime": "2021-03-11 00:27:57.746",
      "Count": 100,
      "Value": 0
    }
  ]
}
```

GET status of model run list

Get status of model runs by run digest, run stamp or run name. If there is only single run with such stamp or name exist then result similar to the result of GET status of model run method.

Methods:

```
GET /api/model/:model/run/:run/status/list
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use run name, which is more human readable than digest.

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Default-4/status/list
http://localhost:4040/api/model/modelOne/run/05403de52f30f59b050417561914fbb8/status/list
http://localhost:4040/api/model/modelOne/run/2019_01_17_19_59_52_998/status/list
```

Return example: *This is a beta version and may change in the future.*

```
[
  {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "Default-4",
    "SubCount": 4,
    "SubStarted": 4,
    "SubCompleted": 4,
    "CreateDateTime": "2021-03-11 00:27:57.119",
    "Status": "s",
    "UpdateDateTime": "2021-03-11 00:27:57.955",
    "RunDigest": "c6ced1efa64dca8a98e5cd323ac7f50d",
    "ValueDigest": "d900353af61f7f824ddae66b47b456ea",
    "RunStamp": "2021_03_11_00_27_57_080",
    "Txt": [],
    "Opts": {},
    "Param": [],
    "Table": [],
    "Progress": [
      {
        "SubId": 0,
        "CreateDateTime": "2021-03-11 00:27:57.151",
        "Status": "s",
        "UpdateDateTime": "2021-03-11 00:27:57.512",
        "Count": 100,
        "Value": 0
      },
      {
        "SubId": 1,
        "CreateDateTime": "2021-03-11 00:27:57.153",
        "Status": "s",
        "UpdateDateTime": "2021-03-11 00:27:57.669",
        "Count": 100,
        "Value": 0
      },
      {
        "SubId": 2,
        "CreateDateTime": "2021-03-11 00:27:57.157",
        "Status": "s",
        "UpdateDateTime": "2021-03-11 00:27:57.649",
        "Count": 100,
        "Value": 0
      },
      {
        "SubId": 3,
        "CreateDateTime": "2021-03-11 00:27:57.159",
        "Status": "s",
        "UpdateDateTime": "2021-03-11 00:27:57.746",
        "Count": 100,
        "Value": 0
      }
    ]
  }
]
```

GET status of first model run

Get status of first model run.

Methods:

```
GET /api/model/:model/run/status/first
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

Call examples:

```
http://localhost:4040/api/model/modelOne/run/status/first
http://localhost:4040/api/model/_201208171604590148_/run/status/first
```

Return example: *This is a beta version and may change in the future.*


```
{
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "Default",
  "SubCount": 1,
  "SubStarted": 1,
  "SubCompleted": 1,
  "CreateDateTime": "2021-03-11 00:27:56.583",
  "Status": "s",
  "UpdateDateTime": "2021-03-11 00:27:57.030",
  "RunDigest": "88b8c45b77993133b07a7c85e4447d5c",
  "ValueDigest": "6c5c0f48e19f67899c868688bb8a23fd",
  "RunStamp": "2021_03_11_00_27_56_535",
  "Txt": [],
  "Opts": {},
  "Param": [],
  "Table": [],
  "Progress": [
    {
      "SubId": 0,
      "CreateDateTime": "2021-03-11 00:27:56.647",
      "Status": "s",
      "UpdateDateTime": "2021-03-11 00:27:56.816",
      "Count": 100,
      "Value": 0
    }
  ]
}
```

GET status of last model run

Get status of last model run.

Methods:

```
GET /api/model/:model/run/status/last
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

Call examples:

```
http://localhost:4040/api/model/modelOne/run/status/last
http://localhost:4040/api/model/_201208171604590148_/run/status/last
```

Return example: *This is a beta version and may change in the future.*


```
{
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "Task Run with NotSuppressed Tables_modelOne_other",
  "SubCount": 2,
  "SubStarted": 2,
  "SubCompleted": 2,
  "CreateDateTime": "2021-03-11 00:28:03.036",
  "Status": "s",
  "UpdateDateTime": "2021-03-11 00:28:03.372",
  "RunDigest": "a5b56959d3f3efd82e7702289af43022",
  "ValueDigest": "79c55110928e7d372c0570cfa2202867",
  "RunStamp": "2021_03_11_00_28_02_520",
  "Txt": [],
  "Opts": {},
  "Param": [],
  "Table": [],
  "Progress": [
    {
      "SubId": 0,
      "CreateDateTime": "2021-03-11 00:28:03.070",
      "Status": "s",
      "UpdateDateTime": "2021-03-11 00:28:03.204",
      "Count": 100,
      "Value": 0
    },
    {
      "SubId": 1,
      "CreateDateTime": "2021-03-11 00:28:03.073",
      "Status": "s",
      "UpdateDateTime": "2021-03-11 00:28:03.195",
      "Count": 100,
      "Value": 0
    }
  ]
}
```

GET status of last completed model run

Get status of last completed model run. Run completed if run status one of: s=success, x=exit, e=error

Methods:

```
GET /api/model/:model/run/status/last-completed
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

Call examples:

```
http://localhost:4040/api/model/modelOne/run/status/last-completed
http://localhost:4040/api/model/_201208171604590148_/run/status/last-completed
```

Return example: *This is a beta version and may change in the future.*


```
{
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "Task Run with NotSuppressed Tables_modelOne_other",
  "SubCount": 2,
  "SubStarted": 2,
  "SubCompleted": 2,
  "CreateDateTime": "2021-03-11 00:28:03.036",
  "Status": "s",
  "UpdateDateTime": "2021-03-11 00:28:03.372",
  "RunDigest": "a5b56959d3f3efd82e7702289af43022",
  "ValueDigest": "79c55110928e7d372c0570cfa2202867",
  "RunStamp": "2021_03_11_00_28_02_520",
  "Txt": [],
  "Opts": {},
  "Param": [],
  "Table": [],
  "Progress": [
    {
      "SubId": 0,
      "CreateDateTime": "2021-03-11 00:28:03.070",
      "Status": "s",
      "UpdateDateTime": "2021-03-11 00:28:03.204",
      "Count": 100,
      "Value": 0
    },
    {
      "SubId": 1,
      "CreateDateTime": "2021-03-11 00:28:03.073",
      "Status": "s",
      "UpdateDateTime": "2021-03-11 00:28:03.195",
      "Count": 100,
      "Value": 0
    }
  ]
}
```

GET model run metadata and status

Get model run results metadata and status

Methods:

```
GET /api/model/:model/run/:run
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Default-4
http://localhost:4040/api/model/_201208171604590148_/run/Default-4
http://localhost:4040/api/model/modelOne/run/2019_01_17_19_59_52_998
```

Return example:


```
{
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "Default-4",
  "SubCount": 4,
  "SubStarted": 4,
  "SubCompleted": 4,
  "CreateDateTime": "2021-11-10 19:08:00.578",
  "Status": "s",
  "UpdateDateTime": "2021-11-10 19:08:04.632",
  "RunDigest": "77074b15c611d2330acc249286bddc04",
  "ValueDigest": "3da3d883d9cb45d419847d3b20cbb6e2",
  "RunStamp": "2021_11_10_19_08_00_552",
  "Txt": [],
  "Opts": {
    "OpenM.LogFilePath": "modelOne.log",
    "OpenM.RunName": "Default-4",
    "OpenM.RunStamp": "2021_11_10_19_08_00_552",
    "OpenM.SetId": "2",
    "OpenM.SetName": "Default",
    "OpenM.SubValues": "4",
    "OpenM.Threads": "4"
  },
  "Param": [
    {
      "Name": "ageSex",
      "Txt": [],
      "SubCount": 1,
      "DefaultSubId": 0,
      "ValueDigest": "ca3edf7630fae786c75f10781a664933"
    },
    {
      "Name": "salaryAge",
      "Txt": [],
      "SubCount": 1,
      "DefaultSubId": 0,
      "ValueDigest": "0becae6201e424a1f3b66e421864b4b3"
    },
    {
      "Name": "StartingSeed",
      "Txt": [],
      "SubCount": 1,
      "DefaultSubId": 0,
      "ValueDigest": "cb565f810da2b25939d0bd958cb5392a"
```

```
    },
    {
      "Name": "salaryFull",
      "Txt": [],
      "SubCount": 1,
      "DefaultSubId": 0,
      "ValueDigest": "a2f1ce089553caf3f7fb080aa170507d"
    },
    {
      "Name": "baseSalary",
      "Txt": [],
      "SubCount": 1,
      "DefaultSubId": 0,
      "ValueDigest": "1541b570479f12a40b9d8a782795c7c2"
    },
    {
      "Name": "filePath",
      "Txt": [],
      "SubCount": 1,
      "DefaultSubId": 0,
      "ValueDigest": "f5d536e282b0941dc84f17cc11a94091"
    },
    {
      "Name": "isOldAge",
      "Txt": [],
      "SubCount": 1,
      "DefaultSubId": 0,
      "ValueDigest": "ef4288d0277e97b1b8a2009ce962323b"
    },
    {
      "Name": "salaryByYears",
      "Txt": [],
      "SubCount": 1,
      "DefaultSubId": 0,
      "ValueDigest": "41934eed3ed19a88b3cb346e447f689f"
    },
    {
      "Name": "salaryByPeriod",
      "Txt": [],
      "SubCount": 1,
      "DefaultSubId": 0,
      "ValueDigest": "3871b18ad0ae36bab0a5badd5bcaab6f"
    },
    {
      "Name": "salaryByLow",
      "Txt": [],
      "SubCount": 1,
      "DefaultSubId": 0,
      "ValueDigest": "b93c3f85f3259f2ad709f39403e7fac9"
    },
    {
      "Name": "salaryByMiddle",
      "Txt": [],
      "SubCount": 1,
      "DefaultSubId": 0,
      "ValueDigest": "6e0d6bf8f96c2d89ff2d2ae2fd82997b"
    }
  ],
  "Table": [
    {
      "Name": "salarySex",
      "ValueDigest": "5b5f5dd270012d1d2eff0d1440613f68"
    },
    {
      "Name": "fullAgeSalary",
      "ValueDigest": "2501fe0596490d69a6e37260f0af35bc"
    },
    {
      "Name": "ageSexIncome",
      "ValueDigest": "aa6c5e76c324cc1bd413afe8e6de6f27"
    },
    {
      "Name": "seedOldAge",
      "ValueDigest": "4883e0ea0adbb4f649ca19aea3b60a78"
    },
    {
      "Name": "incomeByYear",
      "ValueDigest": "83b59f82f2b57268886db6fad85bf423"
    },
    {
      "Name": "incomeByLow",
      "ValueDigest": "d4fac571f0a6943afb96ab428ac79b4a"
    },
    {
      "Name": "incomeByMiddle",
      "ValueDigest": "818c1b6a7ee16d13377e6ffb5355948f"
    },
    {
```

```
      "Name": "incomeByPeriod",
      "ValueDigest": "5379aabc2d6ca654c6e28766ca597d20"
    }
  ],
  "Progress": [
    {
      "SubId": 0,
      "CreateDateTime": "2021-11-10 19:08:04.470",
      "Status": "s",
      "UpdateDateTime": "2021-11-10 19:08:04.552",
      "Count": 100,
      "Value": 0
    },
    {
      "SubId": 1,
      "CreateDateTime": "2021-11-10 19:08:04.470",
      "Status": "s",
      "UpdateDateTime": "2021-11-10 19:08:04.522",
      "Count": 100,
      "Value": 0
    },
    {
      "SubId": 2,
      "CreateDateTime": "2021-11-10 19:08:04.471",
      "Status": "s",
      "UpdateDateTime": "2021-11-10 19:08:04.538",
      "Count": 100,
      "Value": 0
    },
    {
      "SubId": 3,
      "CreateDateTime": "2021-11-10 19:08:04.471",
      "Status": "s",
      "UpdateDateTime": "2021-11-10 19:08:04.570",
      "Count": 100,
      "Value": 0
    }
  ]
}
```

GET model run including text (description and notes)

Get model run results, including text (description and notes)

Methods:

```
GET /api/model/:model/run/:run/text
GET /api/model/:model/run/:run/text/lang/:lang
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

```
:lang - (optional) language code
```

If optional `lang` argument specified then result in that language else in browser language. If no such language exist then text portion of result (description and notes) is empty.

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Default-4/text
http://localhost:4040/api/model/_201208171604590148_/run/Default-4/text
http://localhost:4040/api/model/modelOne/run/Default-4/text/lang/en
http://localhost:4040/api/model/modelOne/run/05403de52f30f59b050417561914fbb8/text/lang/en
http://localhost:4040/api/model/modelOne/run/2019_01_17_19_59_52_998/text/lang/en
```

Return example:


```
{
 "ModelName": "modelOne",
 "ModelDigest": "_201208171604590148_",
 "Name": "Default-4",
 "SubCount": 4,
 "SubStarted": 4,
 "SubCompleted": 4,
 "CreateDateTime": "2021-11-10 19:08:00.578",
 "Status": "s",
 "UpdateDateTime": "2021-11-10 19:08:04.632",
 "RunDigest": "77074b15c611d2330acc249286bddc04",
 "ValueDigest": "3da3d883d9cb45d419847d3b20cbb6e2",
 "RunStamp": "2021_11_10_19_08_00_552",
 "Txt": [
  {
   "LangCode": "EN",
   "Descr": "Model One default set of parameters",
   "Note": ""
  }
 ],
 "Opts": {
  "OpenM.LogFilePath": "modelOne.log",
  "OpenM.RunName": "Default-4",
  "OpenM.RunStamp": "2021_11_10_19_08_00_552",
  "OpenM.SetId": "2",
  "OpenM.SetName": "Default",
  "OpenM.SubValues": "4",
  "OpenM.Threads": "4"
 },
 "Param": [
  {
   "Name": "ageSex",
   "Txt": [
    {
```

```
      "LangCode": "EN",
      "Note": "Age by Sex default values"
    }
  ],
  "SubCount": 1,
  "DefaultSubId": 0,
  "ValueDigest": "ca3edf7630fae786c75f10781a664933"
},
{
  "Name": "salaryAge",
  "Txt": [
    {
      "LangCode": "EN",
      "Note": "Salary by Age default values"
    }
  ],
  "SubCount": 1,
  "DefaultSubId": 0,
  "ValueDigest": "0becae6201e424a1f3b66e421864b4b3"
},
{
  "Name": "StartingSeed",
  "Txt": [
    {
      "LangCode": "EN",
      "Note": "Starting seed default value"
    }
  ],
  "SubCount": 1,
  "DefaultSubId": 0,
  "ValueDigest": "cb565f810da2b25939d0bd958cb5392a"
},
{
  "Name": "salaryFull",
  "Txt": [
    {
      "LangCode": "EN",
      "Note": "Full or part time by Salary default values"
    }
  ],
  "SubCount": 1,
  "DefaultSubId": 0,
  "ValueDigest": "a2f1ce089553caf3f7fb080aa170507d"
},
{
  "Name": "baseSalary",
  "Txt": [],
  "SubCount": 1,
  "DefaultSubId": 0,
  "ValueDigest": "1541b570479f12a40b9d8a782795c7c2"
},
{
  "Name": "filePath",
  "Txt": [],
  "SubCount": 1,
  "DefaultSubId": 0,
  "ValueDigest": "f5d536e282b0941dc84f17cc11a94091"
},
{
  "Name": "isOldAge",
  "Txt": [
    {
      "LangCode": "EN",
      "Note": "Is old age default values"
    }
  ],
  "SubCount": 1,
  "DefaultSubId": 0,
  "ValueDigest": "ef4288d0277e97b1b8a2009ce962323b"
},
{
  "Name": "salaryByYears",
  "Txt": [
    {
      "LangCode": "EN",
      "Note": "Salary by Years default values"
    }
  ],
  "SubCount": 1,
  "DefaultSubId": 0,
  "ValueDigest": "41934eed3ed19a88b3cb346e447f689f"
},
{
  "Name": "salaryByPeriod",
  "Txt": [
    {
      "LangCode": "EN",
```

```
        "Note": "Salary by Period default values"
      }
    ],
    "SubCount": 1,
    "DefaultSubId": 0,
    "ValueDigest": "3871b18ad0ae36bab0a5badd5bcaab6f"
  },
  {
    "Name": "salaryByLow",
    "Txt": [
      {
        "LangCode": "EN",
        "Note": "Salary by Low Period default values"
      }
    ],
    "SubCount": 1,
    "DefaultSubId": 0,
    "ValueDigest": "b93c3f85f3259f2ad709f39403e7fac9"
  },
  {
    "Name": "salaryByMiddle",
    "Txt": [
      {
        "LangCode": "EN",
        "Note": "Salary by Middle Period default values"
      }
    ],
    "SubCount": 1,
    "DefaultSubId": 0,
    "ValueDigest": "6e0d6bf8f96c2d89ff2d2ae2fd82997b"
  }
],
"Table": [
  {
    "Name": "salarySex",
    "ValueDigest": "5b5f5dd270012d1d2eff0d1440613f68"
  },
  {
    "Name": "fullAgeSalary",
    "ValueDigest": "2501fe0596490d69a6e37260f0af35bc"
  },
  {
    "Name": "ageSexIncome",
    "ValueDigest": "aa6c5e76c324cc1bd413afe8e6de6f27"
  },
  {
    "Name": "seedOldAge",
    "ValueDigest": "4883e0ea0adbb4f649ca19aea3b60a78"
  },
  {
    "Name": "incomeByYear",
    "ValueDigest": "83b59f82f2b57268886db6fad85bf423"
  },
  {
    "Name": "incomeByLow",
    "ValueDigest": "d4fac571f0a6943afb96ab428ac79b4a"
  },
  {
    "Name": "incomeByMiddle",
    "ValueDigest": "818c1b6a7ee16d13377e6ffb5355948f"
  },
  {
    "Name": "incomeByPeriod",
    "ValueDigest": "5379aabc2d6ca654c6e28766ca597d20"
  }
],
"Progress": [
  {
    "SubId": 0,
    "CreateDateTime": "2021-11-10 19:08:04.470",
    "Status": "s",
    "UpdateDateTime": "2021-11-10 19:08:04.552",
    "Count": 100,
    "Value": 0
  },
  {
    "SubId": 1,
    "CreateDateTime": "2021-11-10 19:08:04.470",
    "Status": "s",
    "UpdateDateTime": "2021-11-10 19:08:04.522",
    "Count": 100,
    "Value": 0
  },
  {
    "SubId": 2,
    "CreateDateTime": "2021-11-10 19:08:04.471",
    "Status": "s",
    "UpdateDateTime": "2021-11-10 19:08:04.538",
```

```
      "Count": 100,
      "Value": 0
    },
    {
      "SubId": 3,
      "CreateDateTime": "2021-11-10 19:08:04.471",
      "Status": "s",
      "UpdateDateTime": "2021-11-10 19:08:04.570",
      "Count": 100,
      "Value": 0
    }
  ]
}
```

GET model run including text in all languages

Get model run results, including text (description and notes) in all languages

Methods:

```
GET /api/model/:model/run/:run/text/all
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Default/text/all
http://localhost:4040/api/model/_201208171604590148_/run/Default/text/all
http://localhost:4040/api/model/modelOne/run/6fbad822cb9ae42deea1ede626890711/text/all
http://localhost:4040/api/model/modelOne/run/2019_01_17_19_59_52_998/text/all
```

Return example:


```
{
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "Default",
  "SubCount": 1,
  "SubStarted": 1,
  "SubCompleted": 1,
  "CreateDateTime": "2021-11-10 19:07:56.864",
  "Status": "s",
  "UpdateDateTime": "2021-11-10 19:08:00.537",
  "RunDigest": "b2b0878d5b6740983429a06cd856a9b0",
  "ValueDigest": "a1c9a056f2ee40fcc1e07471097845a7",
  "RunStamp": "2021_11_10_19_07_56_837",
  "Txt": [
    {
      "LangCode": "EN",
      "Descr": "Model One default set of parameters",
      "Note": ""
    },
    {
      "LangCode": "FR",
      "Descr": "Modèle Un ensemble de paramètres par défaut",
      "Note": ""
    }
  ],
  "Opts": {
    "OpenM.LogFilePath": "modelOne.log",
    "OpenM.RunName": "Default",
    "OpenM.RunStamp": "2021_11_10_19_07_56_837",
    "OpenM.SetId": "2",
    "OpenM.SetName": "Default"
  },
  "Param": [
    {
      "Name": "ageSex",
      "Txt": [
        {
          "LangCode": "EN",
          "Note": "Age by Sex default values"
        },
        {
          "LangCode": "FR",
          "Note": "Valeurs par défaut de l'Âge par Sexe"
```

```
      }
    ],
    "SubCount": 1,
    "DefaultSubId": 0,
    "ValueDigest": "ca3edf7630fae786c75f10781a664933"
  },
  {
    "Name": "salaryAge",
    "Txt": [
      {
        "LangCode": "EN",
        "Note": "Salary by Age default values"
      },
      {
        "LangCode": "FR",
        "Note": "Salaire par Âge valeurs par défaut"
      }
    ],
    "SubCount": 1,
    "DefaultSubId": 0,
    "ValueDigest": "0becae6201e424a1f3b66e421864b4b3"
  },
  {
    "Name": "StartingSeed",
    "Txt": [
      {
        "LangCode": "EN",
        "Note": "Starting seed default value"
      }
    ],
    "SubCount": 1,
    "DefaultSubId": 0,
    "ValueDigest": "cb565f810da2b25939d0bd958cb5392a"
  },
  {
    "Name": "salaryFull",
    "Txt": [
      {
        "LangCode": "EN",
        "Note": "Full or part time by Salary default values"
      }
    ],
    "SubCount": 1,
    "DefaultSubId": 0,
    "ValueDigest": "a2f1ce089553caf3f7fb080aa170507d"
  },
  {
    "Name": "baseSalary",
    "Txt": [],
    "SubCount": 1,
    "DefaultSubId": 0,
    "ValueDigest": "1541b570479f12a40b9d8a782795c7c2"
  },
  {
    "Name": "filePath",
    "Txt": [],
    "SubCount": 1,
    "DefaultSubId": 0,
    "ValueDigest": "f5d536e282b0941dc84f17cc11a94091"
  },
  {
    "Name": "isOldAge",
    "Txt": [
      {
        "LangCode": "EN",
        "Note": "Is old age default values"
      }
    ],
    "SubCount": 1,
    "DefaultSubId": 0,
    "ValueDigest": "ef4288d0277e97b1b8a2009ce962323b"
  },
  {
    "Name": "salaryByYears",
    "Txt": [
      {
        "LangCode": "EN",
        "Note": "Salary by Years default values"
      }
    ],
    "SubCount": 1,
    "DefaultSubId": 0,
    "ValueDigest": "41934eed3ed19a88b3cb346e447f689f"
  },
  {
    "Name": "salaryByPeriod",
    "Txt": [
      {
```

```
        {
          "LangCode": "EN",
          "Note": "Salary by Period default values"
        }
      ],
      "SubCount": 1,
      "DefaultSubId": 0,
      "ValueDigest": "3871b18ad0ae36bab0a5badd5bcaab6f"
    },
    {
      "Name": "salaryByLow",
      "Txt": [
        {
          "LangCode": "EN",
          "Note": "Salary by Low Period default values"
        }
      ],
      "SubCount": 1,
      "DefaultSubId": 0,
      "ValueDigest": "b93c3f85f3259f2ad709f39403e7fac9"
    },
    {
      "Name": "salaryByMiddle",
      "Txt": [
        {
          "LangCode": "EN",
          "Note": "Salary by Middle Period default values"
        }
      ],
      "SubCount": 1,
      "DefaultSubId": 0,
      "ValueDigest": "6e0d6bf8f96c2d89ff2d2ae2fd82997b"
    }
  ],
  "Table": [
    {
      "Name": "salarySex",
      "ValueDigest": "2d860e00b49881ed802377529236fc0e"
    },
    {
      "Name": "fullAgeSalary",
      "ValueDigest": "6f55fbd529a126a6d5ac6a6e855476ce"
    },
    {
      "Name": "ageSexIncome",
      "ValueDigest": "72121007312255cdcad7a82b46e6aa9c"
    },
    {
      "Name": "seedOldAge",
      "ValueDigest": "df4c82301d470072348f996b7d75424d"
    },
    {
      "Name": "incomeByYear",
      "ValueDigest": "83b59f82f2b57268886db6fad85bf423"
    },
    {
      "Name": "incomeByLow",
      "ValueDigest": "d4fac571f0a6943afb96ab428ac79b4a"
    },
    {
      "Name": "incomeByMiddle",
      "ValueDigest": "818c1b6a7ee16d13377e6ffb5355948f"
    },
    {
      "Name": "incomeByPeriod",
      "ValueDigest": "5379aabc2d6ca654c6e28766ca597d20"
    }
  ],
  "Progress": [
    {
      "SubId": 0,
      "CreateDateTime": "2021-11-10 19:08:00.399",
      "Status": "s",
      "UpdateDateTime": "2021-11-10 19:08:00.433",
      "Count": 100,
      "Value": 0
    }
  ]
}
```

GET list of model worksets

Get list of model worksets: language-neutral part of workset list metadata. Workset is a set of model input parameters (a.k.a. "scenario" input). Workset can be used to run the model.

Methods:

```
GET /api/model/:model/workset-list
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

Call examples:

```
http://localhost:4040/api/model/modelOne/workset-list
http://localhost:4040/api/model/649f17f26d67c37b78dde94f79772445/workset-list
```

Return example:


```
[
  {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "modelOne",
    "BaseRunDigest": "",
    "IsReadonly": true,
    "UpdateDateTime": "2013-05-29 23:55:07.1234",
    "Txt": [],
    "Param": []
  },
  {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "modelOne_set",
    "BaseRunDigest": "",
    "IsReadonly": false,
    "UpdateDateTime": "2013-05-30 23:55:07.1234",
    "Txt": [],
    "Param": []
  },
  {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "modelOne_other",
    "BaseRunDigest": "",
    "IsReadonly": true,
    "UpdateDateTime": "2013-05-29 23:55:07.1234",
    "Txt": [],
    "Param": []
  }
]
```

GET list of model worksets including text (description and notes)

Get list of model worksets, including text (description and notes). Workset is a set of model input parameters (a.k.a. "scenario" input). Workset can be used to run the model.

Methods:

```
GET /api/model/:model/workset-list/text
GET /api/model/:model/workset-list/text/lang/:lang
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:lang - (optional) language code
```

If optional `lang` argument specified then result in that language else in browser language. If no such language exist then text portion of result (description and notes) is empty.

Call examples:

```
http://localhost:4040/api/model/modelOne/workset-list/text
http://localhost:4040/api/model/649f17f26d67c37b78dde94f79772445/workset-list/text
http://localhost:4040/api/model/modelOne/workset-list/text/lang/fr-FR
```

Return example:

```
[
  {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "modelOne",
    "BaseRunDigest": "",
    "IsReadonly": true,
    "UpdateDateTime": "2013-05-29 23:55:07.1234",
    "Txt": [
      {
        "LangCode": "EN",
        "Descr": "Model One default set of parameters",
        "Note": ""
      }
    ],
    "Param": []
  },
  {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "modelOne_set",
    "BaseRunDigest": "",
    "IsReadonly": false,
    "UpdateDateTime": "2013-05-30 23:55:07.1234",
    "Txt": [
      {
        "LangCode": "EN",
        "Descr": "modelOne modified set of parameters",
        "Note": ""
      }
    ],
    "Param": []
  },
  {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "modelOne_other",
    "BaseRunDigest": "",
    "IsReadonly": true,
    "UpdateDateTime": "2013-05-29 23:55:07.1234",
    "Txt": [
      {
        "LangCode": "EN",
        "Descr": "Model One other set of parameters",
        "Note": ""
      }
    ],
    "Param": []
  }
]
```

GET workset status

Get status of model workset. Workset is a set of model input parameters (a.k.a. "scenario" input). Workset can be used to run the model.

Methods:

```
GET /api/model/:model/workset/:set/status
GET /api/model/:model/workset/:set
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:set - (required) workset name
```

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default".

Call examples:

```
http://localhost:4040/api/model/modelOne/workset/modelOne_set/status
http://localhost:4040/api/model/649f17f26d67c37b78dde94f79772445/workset/Default/status
```

Return example: *This is a beta version and may change in the future.*


```
{
  "SetId": 101,
  "BaseRunId": 0,
  "ModelId": 101,
  "Name": "Default",
  "IsReadonly": true,
  "UpdateDateTime": "2017-12-19 15:21:14.0232"
}
```

GET model default workset status

Get status of default model workset. Workset is a set of model input parameters (a.k.a. "scenario" input). Workset can be used to run the model. Default workset is a first workset of the model with set_id = min(set_id) for that model.

Methods:

```
GET /api/model/:model/workset/status/default
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

Call examples:

```
http://localhost:4040/api/model/modelOne/workset/status/default
http://localhost:4040/api/model/649f17f26d67c37b78dde94f79772445/workset/status/default
```

Return example: *This is a beta version and may change in the future.*


```
{
  "SetId": 101,
  "BaseRunId": 0,
  "ModelId": 101,
  "Name": "Default",
  "IsReadonly": true,
  "UpdateDateTime": "2017-12-19 15:21:14.0232"
}
```

GET workset including text (description and notes)

Get model workset metadata, including text (description and notes). Workset is a set of model input parameters (a.k.a. "scenario" input). Workset can be used to run the model.

Methods:

```
GET /api/model/:model/workset/:set/text
GET /api/model/:model/workset/:set/text/lang/:lang
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:set - (required) workset name
```

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default".

```
:lang - (optional) language code
```

If optional `lang` argument specified then result in that language else in browser language. If no such language exist then text portion of result (description and notes) is empty.

Call examples:

```
http://localhost:4040/api/model/modelOne/workset/modelOne_set/text
http://localhost:4040/api/model/649f17f26d67c37b78dde94f79772445/workset/Default/text
http://localhost:4040/api/model/modelOne/workset/modelOne_set/text/lang/FR
http://localhost:4040/api/model/649f17f26d67c37b78dde94f79772445/workset/Default/text/lang/en
```

Return example:

```
{
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "Default",
  "BaseRunDigest": "",
  "IsReadonly": true,
  "UpdateDateTime": "2020-03-17 12:10:48.303",
  "Txt": [
    {
      "LangCode": "EN",
      "Descr": "Model One default set of parameters",
      "Note": ""
    }
  ],
  "Param": [
    {
      "Name": "ageSex",
      "SubCount": 1,
      "DefaultSubId": 0,
      "Txt": [
        {
          "LangCode": "EN",
          "Note": "Age by Sex default values"
        }
      ]
    },
    {
      "Name": "salaryAge",
      "SubCount": 1,
      "DefaultSubId": 0,
      "Txt": [
        {
          "LangCode": "EN",
          "Note": "Salary by Age default values"
        }
      ]
    },
    {
      "Name": "StartingSeed",
      "SubCount": 1,
      "DefaultSubId": 0,
      "Txt": [
        {
          "LangCode": "EN",
          "Note": "Starting seed default value"
        }
      ]
    },
    {
      "Name": "salaryFull",
      "SubCount": 4,
      "DefaultSubId": 3,
      "Txt": [
        {
          "LangCode": "EN",
          "Note": "Full or part time by Salary default values"
        }
      ]
    },
    {
      "Name": "baseSalary",
      "SubCount": 4,
      "DefaultSubId": 3,
      "Txt": []
    },
    {
      "Name": "filePath",
      "SubCount": 4,
      "DefaultSubId": 3,
      "Txt": []
    },
    {
      "Name": "isOldAge",
      "SubCount": 4,
      "DefaultSubId": 3,
      "Txt": [
        {
          "LangCode": "EN",
          "Note": "Is old age default values"
        }
      ]
    }
  ]
}
```

GET workset including text in all languages

Get model workset metadata, including text (description and notes), in all languages. Workset is a set of model input parameters (a.k.a. "scenario" input). Workset can be used to run the model.

Methods:

```
GET /api/model/:model/workset/:set/text/all
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:set - (required) workset name
```

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default".

Call examples:

```
http://localhost:4040/api/model/modelOne/workset/modelOne_set/text/all
http://localhost:4040/api/model/649f17f26d67c37b78dde94f79772445/workset/Default/text/all
```

Return example:


```
{
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "Default",
  "BaseRunDigest": "",
  "IsReadonly": true,
  "UpdateDateTime": "2021-09-22 21:37:46.792",
  "Txt": [
    {
      "LangCode": "EN",
      "Descr": "Model One default set of parameters",
      "Note": ""
    },
    {
      "LangCode": "FR",
      "Descr": "(FR) Model One default set of parameters",
      "Note": ""
    }
  ],
  "Param": [
    {
      "Name": "ageSex",
      "SubCount": 1,
      "DefaultSubId": 0,
      "Txt": [
        {
          "LangCode": "EN",
          "Note": "Age by Sex default values"
        },
        {
          "LangCode": "FR",
          "Note": "(FR) Age by Sex default values"
        }
      ]
    },
    {
      "Name": "salaryAge",
      "SubCount": 1,
      "DefaultSubId": 0,
      "Txt": [
        {
          "LangCode": "EN",
          "Note": "Salary by Age default values"
        },
        {
          "LangCode": "FR",
          "Note": "(FR) Salary by Age default values"
```

```
        "Note": "(FT) Salary by Age default values"
      }
    ]
  },
  {
    "Name": "StartingSeed",
    "SubCount": 1,
    "DefaultSubId": 0,
    "Txt": [
      {
        "LangCode": "EN",
        "Note": "Starting seed default value"
      }
    ]
  },
  {
    "Name": "salaryFull",
    "SubCount": 4,
    "DefaultSubId": 3,
    "Txt": [
      {
        "LangCode": "EN",
        "Note": "Full or part time by Salary default values"
      }
    ]
  },
  {
    "Name": "baseSalary",
    "SubCount": 4,
    "DefaultSubId": 3,
    "Txt": []
  },
  {
    "Name": "filePath",
    "SubCount": 4,
    "DefaultSubId": 3,
    "Txt": []
  },
  {
    "Name": "isOldAge",
    "SubCount": 4,
    "DefaultSubId": 3,
    "Txt": [
      {
        "LangCode": "EN",
        "Note": "Is old age default values"
      }
    ]
  }
 ]
}
```

Read parameter values from workset

Read a "page" of parameter values from workset.

Page is part of parameter values defined by zero-based "start" row number and row count. If row count <= 0 then all rows below start row number returned.

Dimension(s) and enum-based parameters returned as enum codes. If dimension type or parameter type is simple (integer or boolean) then string value used (ex.: "true", "1234").

Method verb must be POST and Content-Type header "application/json". JSON body POSTed to specify parameter name, page size, row count, filters and row order. It is expected to be JSON representation of db.ReadLayout structure from Go library.

Method:

```
POST /api/model/:model/workset/:set/parameter/value
```

Call example:

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/workset/Default/parameter/value -d @test.json
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:set - (required) workset name
```

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default".

JSON body arguments:

For example:

```
{
  "Name": "ageSex",
  "Offset": 0,
  "Size": 100,
  "IsLastPage": true,
  "Filter": [{
      "Name": "dim0",
      "Op": "IN",
      "Values": ["20-30", "40+"]
    }, {
      "Name": "dim1",
      "Op": "=",
      "Values": ["F"]
    }
  ],
  "OrderBy": [{
      "IndexOne": 2,
      "IsDesc": true
    }, {
      "IndexOne": 3,
      "IsDesc": true
    }
  ]
}
```

```
Name       - (required) parameter name
Offset     - (optional) zero-based start row to select parameter values
Size       - (optional) max row count to select parameter values.
IsLastPage - (optional) if true then always return non-empty last page of data.
Filter     - (optional) conditions to filter dimension enum code(s)
OrderBy    - (optional) list of columns indexes (one based) to order by
```

By default oms service selects 100 rows (it can be configured). If `Size` <= 0 specified then all rows selected.

Filter conditions joined by AND and can have following operations:

```
=       - enum equal to:          AgeGroup = "20-30"
!=      - enum not equal to:      AgeGroup <> "20-30"
>       - enum greater than:      AgeGroup > "20-30"
>=      - enum greater or equal:  AgeGroup >= "20-30"
<       - enum less than:         AgeGroup < "20-30"
<=      - enum less or equal:     AgeGroup <= "20-30"
IN      - enum is in the list of: AgeGroup IN ("20-30", "30-40", "40+")
BETWEEN - between min and max:    AgeGroup BETWEEN "30-40" AND "all"
IN_AUTO - automatically choose most suitable: = or != or IN or BETWEEN
```

Keep in mind: dimension enums are always ordered by id's, not by code and result of filter `Sex < "M"` may not be `Sex = "F"`.

Order by specified by one-based column(s) index(es) in result. In case of parameters columns are:

```
  SELECT sub_id, dim0, dim1, ..., value FROM parameterTable ORDER BY 1, 2,...
```

Columns always contain enum id's, not enum codes and therefore result ordered by id's

JSON response:

```
{
  Layout: {
    Offset:     actual first row number of the page data (zero-base),
    Size:       actual data page row count,
    IsLastPage: true if this is last page of data
  },
  Page: [....page of data...]
}
```

Example 1:

JSON body:

```
{
  "Name": "ageSex",
  "Filter": [],
  "OrderBy": []
}
```

Result:

```
< HTTP/1.1 200 OK
< Access-Control-Allow-Origin: *
< Content-Type: application/json
< Date: Tue, 19 Dec 2017 17:13:51 GMT
< Content-Length: 424
<
{"Layout":{"Offset":0,"Size":8,"IsLastPage":true}
,"Page":[{"Dims":["10-20","M"],"IsNull":false,"Value":0.1,"SubId":0}
,{"Dims":["10-20","F"],"IsNull":false,"Value":0.2,"SubId":0}
,{"Dims":["20-30","M"],"IsNull":false,"Value":0.3,"SubId":0}
,{"Dims":["20-30","F"],"IsNull":false,"Value":0.4,"SubId":0}
,{"Dims":["30-40","M"],"IsNull":false,"Value":0.5,"SubId":0}
,{"Dims":["30-40","F"],"IsNull":false,"Value":0.6,"SubId":0}
,{"Dims":["40+","M"],"IsNull":false,"Value":0.7,"SubId":0}
,{"Dims":["40+","F"],"IsNull":false,"Value":0.8,"SubId":0}
]}
```

Example 2:

JSON body:

```
{
  "Name":       "ageSex",
  "Offset":     6,
  "Size":       4,
  "IsLastPage": true,
  "Filter":     [],
  "OrderBy":    []
}
```

Result:

```
{"Layout":{"Offset":6,"Size":2,"IsLastPage":true}
,"Page":[{"Dims":["40+","M"],"IsNull":false,"Value":0.7,"SubId":0}
,{"Dims":["40+","F"],"IsNull":false,"Value":0.8,"SubId":0}
]}
```

Example 3:

JSON body:

```
{
  "Name": "ageSex",
  "Offset": 2,
  "OrderBy": [{
      "IndexOne": 2,
      "IsDesc": true
    }, {
      "IndexOne": 3,
      "IsDesc": true
    }
  ]
}
```

Result:

```
{"Layout":{"Offset":2,"Size":6,"IsLastPage":true}
,"Page":[{"Dims":["30-40","F"],"IsNull":false,"Value":0.6,"SubId":0}
,{"Dims":["30-40","M"],"IsNull":false,"Value":0.5,"SubId":0}
,{"Dims":["20-30","F"],"IsNull":false,"Value":0.4,"SubId":0}
,{"Dims":["20-30","M"],"IsNull":false,"Value":0.3,"SubId":0}
,{"Dims":["10-20","F"],"IsNull":false,"Value":0.2,"SubId":0}
,{"Dims":["10-20","M"],"IsNull":false,"Value":0.1,"SubId":0}
]}
```

Example 4:

JSON body:

```
{
  "Name": "ageSex",
  "Offset": 0,
  "Size": 100,
  "Filter": [{
      "Name": "dim0",
      "Op": "IN",
      "Values": ["20-30", "40+"]
    }, {
      "Name": "dim1",
      "Op": "=",
      "Values": ["F"]
    }
  ],
  "OrderBy": [{
      "IndexOne": 2,
      "IsDesc": true
    }, {
      "IndexOne": 3,
      "IsDesc": true
    }
  ]
}
```

Result:

{"Layout":{"Offset":0,"Size":2,"IsLastPage":true}
,"Page":[{"Dims":["40+","F"],"IsNull":false,"Value":0.8,"SubId":0}
,{"Dims":["20-30","F"],"IsNull":false,"Value":0.4,"SubId":0}
]}

Read parameter values from workset (enum id's)

Read a "page" of parameter values from workset.

Page is part of parameter values defined by zero-based "start" row number and row count. If row count <= 0 then all rows below start row number returned. Dimension(s) and enum-based parameters returned as enum id, not enum codes.

Method verb must be POST and Content-Type header "application/json". JSON body POSTed to specify parameter name, page size, row count, filters and row order. It is expected to be JSON representation of db.ReadLayout structure from Go library.

Method:

```
POST /api/model/:model/workset/:set/parameter/value-id
```

For example:

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/workset/Default/parameter/value-id -d @test.json
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:set - (required) workset name
```

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default".

JSON body arguments:

For example:

```
{
  "Name": "ageSex",
  "Offset": 0,
  "Size": 100,
  "IsLastPage": true,
  "FilterById": [{
      "Name": "AgeGroup",
      "Op": "IN",
      "EnumIds": [20, 40]
    }, {
      "Name": "Sex",
      "Op": "=",
      "EnumIds": [1]
    }
  ],
  "OrderBy": [{
      "IndexOne": 2,
      "IsDesc": true
    }, {
      "IndexOne": 3,
      "IsDesc": true
    }
  ]
}
```

```
Name       - (required) parameter name
Offset     - (optional) zero-based start row to select parameter values
Size       - (optional) max row count to select parameter values.
IsLastPage - (optional) if true then always return non-empty last page of data.
FilterById - (optional) conditions to filter dimension enum id's
OrderBy    - (optional) list of columns indexes (one based) to order by
```

By default oms service selects 100 rows (it can be configured). If `Size` <= 0 specified then all rows selected.

Filter conditions joined by AND and can have following operations:

```
=       - enum id equal to:          AgeGroup = 20
!=      - enum id not equal to:      AgeGroup <> 20
>       - enum id greater than:      AgeGroup > 20
>=      - enum id greater or equal: AgeGroup >= 20
<       - enum id less than:         AgeGroup < 20
<=      - enum id less or equal:     AgeGroup <= 20
IN      - in the list of id's:       AgeGroup IN (20, 30, 40)
BETWEEN - between min and max:       AgeGroup BETWEEN 20 AND 40
IN_AUTO - automatically choose most suitable: = or != or IN or BETWEEN
```

Order by specified by one-based column(s) index(es) in result. In case of parameters columns are:

```
SELECT sub_id, dim0, dim1, ..., value FROM parameterTable
```

JSON response:

```
{
  Layout: {
    Offset:     actual first row number of the page data (zero-base),
    Size:       actual data page row count,
    IsLastPage: true if this is last page of data
  },
  Page: [....page of data...]
}
```

Example 1:

JSON body:

```
{
  "Name": "ageSex"
}
```

Result:

```
< HTTP/1.1 200 OK
< Content-Type: application/json
< Date: Fri, 14 Dec 2018 01:48:51 GMT
< Content-Length: 508
<
{"Layout":{"Offset":0,"Size":8,"IsLastPage":true}
,"Page":[{"DimIds":[10,0],"IsNull":false,"Value":0.1,"SubId":0}
,{"DimIds":[10,1],"IsNull":false,"Value":0.2,"SubId":0}
,{"DimIds":[20,0],"IsNull":false,"Value":0.3,"SubId":0}
,{"DimIds":[20,1],"IsNull":false,"Value":0.4,"SubId":0}
,{"DimIds":[30,0],"IsNull":false,"Value":0.5,"SubId":0}
,{"DimIds":[30,1],"IsNull":false,"Value":0.6,"SubId":0}
,{"DimIds":[40,0],"IsNull":false,"Value":0.7,"SubId":0}
,{"DimIds":[40,1],"IsNull":false,"Value":0.8,"SubId":0}
]}
```

Example 2:

JSON body:

```
{
  "Name":       "ageSex",
  "Offset":     6,
  "Size":       4,
  "IsLastPage": true
}
```

Result:

```
{"Layout":{"Offset":6,"Size":2,"IsLastPage":true}
,"Page":[{"DimIds":[40,0],"IsNull":false,"Value":0.7,"SubId":0}
,{"DimIds":[40,1],"IsNull":false,"Value":0.8,"SubId":0}
]}
```

Example 3:

JSON body:

```
{
  "Name": "ageSex",
  "Offset": 2,
  "OrderBy": [{
      "IndexOne": 2,
      "IsDesc": true
    }, {
      "IndexOne": 3,
      "IsDesc": true
    }
  ]
}
```

Result:

```
{"Layout":{"Offset":2,"Size":6,"IsLastPage":true}
,"Page":[{"DimIds":[30,1],"IsNull":false,"Value":0.6,"SubId":0}
,{"DimIds":[30,0],"IsNull":false,"Value":0.5,"SubId":0}
,{"DimIds":[20,1],"IsNull":false,"Value":0.4,"SubId":0}
,{"DimIds":[20,0],"IsNull":false,"Value":0.3,"SubId":0}
,{"DimIds":[10,1],"IsNull":false,"Value":0.2,"SubId":0}
,{"DimIds":[10,0],"IsNull":false,"Value":0.1,"SubId":0}
]}
```

Example 4:

JSON body:

```
{
  "Name": "ageSex",
  "Offset": 0,
  "Size": 100,
  "FilterById": [{
      "Name": "dim0",
      "Op": "IN",
      "EnumIds": [20, 40]
    }, {
      "Name": "dim1",
      "Op": "=",
      "EnumIds": [1]
    }
  ],
  "OrderBy": [{
      "IndexOne": 2,
      "IsDesc": true
    }, {
      "IndexOne": 3,
      "IsDesc": true
    }
  ]
}
```

Result:

```
{"Layout":{"Offset":0,"Size":2,"IsLastPage":true}
,"Page":[{"DimIds":[40,1],"IsNull":false,"Value":0.8,"SubId":0}
,{"DimIds":[20,1],"IsNull":false,"Value":0.4,"SubId":0}
]}
```

Read parameter values from model run

Read a "page" of parameter values from model run.

Page is part of parameter values defined by zero-based "start" row number and row count. If row count <= 0 then all rows below start row number returned.

Dimension(s) and enum-based parameters returned as enum codes. Dimension type or parameter type is simple (integer or boolean) then string value used (ex.: "true", "1234").

Method verb must be POST and Content-Type header "application/json". JSON body POSTed to specify parameter name, page size, row count, filters and row order. It is expected to be JSON representation of db.ReadLayout structure from Go library.

Method:

```
POST /api/model/:model/run/:run/parameter/value
```

For example:

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/run/Default/parameter/value -d @test.json
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/run/2016_08_17_21_07_55_123/parameter/value -d @test.json
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

JSON body arguments:

For example:

```
{
  "Name": "ageSex",
  "Offset": 0,
  "Size": 100,
  "IsLastPage": true,
  "Filter": [{
      "Name": "AgeGroup",
      "Op": "IN",
      "Values": ["20-30", "30-40", "40+"]
    }, {
      "Name": "Sex",
      "Op": "=",
      "Values": ["F"]
    }
  ],
  "OrderBy": [{
      "IndexOne": 2,
      "IsDesc": true
    }, {
      "IndexOne": 3,
      "IsDesc": true
    }
  ]
}
```

```
Name       - (required) parameter name
Offset     - (optional) zero-based start row to select parameter values
Size       - (optional) max row count to select parameter values.
IsLastPage - (optional) if true then always return non-empty last page of data.
Filter     - (optional) conditions to filter dimension enum code(s)
OrderBy    - (optional) list of columns indexes (one based) to order by
```

By default oms service selects 100 rows (it can be configured). If `Size` <= 0 specified then all rows selected.

Filter conditions joined by AND and can have following operations:

```
=       - enum equal to:          AgeGroup = "20-30"
!=      - enum not equal to:      AgeGroup <> "20-30"
>       - enum greater than:      AgeGroup > "20-30"
>=      - enum greater or equal:  AgeGroup >= "20-30"
<       - enum less than:         AgeGroup < "20-30"
<=      - enum less or equal:     AgeGroup <= "20-30"
IN      - enum is in the list of: AgeGroup IN ("20-30", "30-40", "40+")
BETWEEN - between min and max:    AgeGroup BETWEEN "30-40" AND "all"
IN_AUTO - automatically choose most suitable: = or != or IN or BETWEEN
```

Keep in mind: dimension enums are always ordered by id's, not by code and result of filter `Sex < "M"` may not be `Sex = "F"`.

Order by specified by one-based column(s) index(es) in result. In case of parameters columns are:

```
  SELECT sub_id, dim0, dim1, ..., value FROM parameterTable ORDER BY 1, 2,...
```

Columns always contain enum id's, not enum codes and therefore result ordered by id's

JSON response:

```
{
  Layout: {
    Offset:     actual first row number of the page data (zero-base),
    Size:       actual data page row count,
    IsLastPage: true if this is last page of data
  },
  Page: [....page of data...]
}
```

Example 1:

JSON body:

```
{
  "Name": "ageSex",
  "Filter": [],
  "OrderBy": []
}
```

Result:

```
< HTTP/1.1 200 OK
< Content-Type: application/json
< Date: Fri, 14 Dec 2018 01:53:21 GMT
< Content-Length: 544
<
{"Layout":{"Offset":0,"Size":8,"IsLastPage":true}
,"Page":[{"Dims":["10-20","M"],"IsNull":false,"Value":0.1,"SubId":0}
,{"Dims":["10-20","F"],"IsNull":false,"Value":0.2,"SubId":0}
,{"Dims":["20-30","M"],"IsNull":false,"Value":0.3,"SubId":0}
,{"Dims":["20-30","F"],"IsNull":false,"Value":0.4,"SubId":0}
,{"Dims":["30-40","M"],"IsNull":false,"Value":0.5,"SubId":0}
,{"Dims":["30-40","F"],"IsNull":false,"Value":0.6,"SubId":0}
,{"Dims":["40+","M"],"IsNull":false,"Value":0.7,"SubId":0}
,{"Dims":["40+","F"],"IsNull":false,"Value":0.8,"SubId":0}
]}
```

Example 2:

JSON body:

```
{
  "Name":      "ageSex",
  "Offset":    6,
  "Size":      4,
  "IsLastPage": true,
  "Filter":    [],
  "OrderBy":   []
}
```

Result:


```
{"Layout":{"Offset":6,"Size":2,"IsLastPage":true}
,"Page":[{"Dims":["40+","M"],"IsNull":false,"Value":0.7,"SubId":0}
,{"Dims":["40+","F"],"IsNull":false,"Value":0.8,"SubId":0}
]}
```


Example 3:

JSON body:

```
{
  "Name": "ageSex",
  "Offset": 2,
  "OrderBy": [{
      "IndexOne": 2,
      "IsDesc": true
    }, {
      "IndexOne": 3,
      "IsDesc": true
    }
  ]
}
```

Result:


```
{"Layout":{"Offset":2,"Size":6,"IsLastPage":true}
,"Page":[{"Dims":["30-40","F"],"IsNull":false,"Value":0.6,"SubId":0}
,{"Dims":["30-40","M"],"IsNull":false,"Value":0.5,"SubId":0}
,{"Dims":["20-30","F"],"IsNull":false,"Value":0.4,"SubId":0}
,{"Dims":["20-30","M"],"IsNull":false,"Value":0.3,"SubId":0}
,{"Dims":["10-20","F"],"IsNull":false,"Value":0.2,"SubId":0}
,{"Dims":["10-20","M"],"IsNull":false,"Value":0.1,"SubId":0}
]}
```


Example 4:

JSON body:

```
{
  "Name": "ageSex",
  "Offset": 0,
  "Size": 100,
  "Filter": [{
      "Name": "dim0",
      "Op": "IN",
      "Values": ["20-30", "40+"]
    }, {
      "Name": "dim1",
      "Op": "=",
      "Values": ["F"]
    }
  ],
  "OrderBy": [{
      "IndexOne": 2,
      "IsDesc": true
    }, {
      "IndexOne": 3,
      "IsDesc": true
    }
  ]
}
```

Result:

{"Layout":{"Offset":0,"Size":2,"IsLastPage":true}
,"Page":[{"Dims":["40+","F"],"IsNull":false,"Value":0.8,"SubId":0}
,{"Dims":["20-30","F"],"IsNull":false,"Value":0.4,"SubId":0}
]}

Read parameter values from model run (enum id's)

Read a "page" of parameter values from model run.

Page is part of parameter values defined by zero-based "start" row number and row count. If row count <= 0 then all rows below start row number returned. Dimension(s) and enum-based parameters returned as enum id, not enum codes.

Method verb must be POST and Content-Type header "application/json". JSON body POSTed to specify parameter name, page size, row count, filters and row order. It is expected to be JSON representation of db.ReadLayout structure from Go library.

Method:

```
POST /api/model/:model/run/:run/parameter/value-id
```

For example:

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/run/Default/parameter/value-id -d @test.json
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/run/2019_01_17_19_59_52_998/parameter/value-id -d @test.json
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

JSON body arguments:

For example:

```
{
  "Name": "ageSex",
  "Offset": 0,
  "Size": 100,
  "IsLastPage": true,
  "FilterById": [{
      "Name": "AgeGroup",
      "Op": "IN",
      "EnumIds": [20, 40]
    }, {
      "Name": "Sex",
      "Op": "=",
      "EnumIds": [1]
    }
  ],
  "OrderBy": [{
      "IndexOne": 2,
      "IsDesc": true
    }, {
      "IndexOne": 3,
      "IsDesc": true
    }
  ]
}
```

```
Name       - (required) parameter name
Offset     - (optional) zero-based start row to select parameter values
Size       - (optional) max row count to select parameter values.
IsLastPage - (optional) if true then always return non-empty last page of data.
FilterById - (optional) conditions to filter dimension enum code(s)
OrderBy    - (optional) list of columns indexes (one based) to order by
```

By default oms service selects 100 rows (it can be configured). If `Size` <= 0 specified then all rows selected.

Filter conditions joined by AND and can have following operations:

```
=       - enum id equal to:         AgeGroup = 20
!=      - enum id not equal to:     AgeGroup <> 20
>       - enum id greater than:     AgeGroup > 20
>=      - enum id greater or equal: AgeGroup >= 20
<       - enum id less than:        AgeGroup < 20
<=      - enum id less or equal:    AgeGroup <= 20
IN      - in the list of id's:      AgeGroup IN (20, 30, 40)
BETWEEN - between min and max:      AgeGroup BETWEEN 20 AND 40
IN_AUTO - automatically choose most suitable: = or != or IN or BETWEEN
```

Order by specified by one-based column(s) index(es) in result. In case of parameters columns are:

```
  SELECT sub_id, dim0, dim1, ..., value FROM parameterTable ORDER BY 1, 2,...
```

Columns always contain enum id's, not enum codes and therefore result ordered by id's

JSON response:

```
{
  Layout: {
    Offset:     actual first row number of the page data (zero-base),
    Size:       actual data page row count,
    IsLastPage: true if this is last page of data
  },
  Page: [....page of data...]
}
```

Example 1:

JSON body:

```
{
  "Name": "ageSex"
}
```

Result:

```
< HTTP/1.1 200 OK
< Content-Type: application/json
< Date: Fri, 14 Dec 2018 01:56:34 GMT
< Content-Length: 508
<
{"Layout":{"Offset":0,"Size":8,"IsLastPage":true}
,"Page":[{"DimIds":[10,0],"IsNull":false,"Value":0.1,"SubId":0}
,{"DimIds":[10,1],"IsNull":false,"Value":0.2,"SubId":0}
,{"DimIds":[20,0],"IsNull":false,"Value":0.3,"SubId":0}
,{"DimIds":[20,1],"IsNull":false,"Value":0.4,"SubId":0}
,{"DimIds":[30,0],"IsNull":false,"Value":0.5,"SubId":0}
,{"DimIds":[30,1],"IsNull":false,"Value":0.6,"SubId":0}
,{"DimIds":[40,0],"IsNull":false,"Value":0.7,"SubId":0}
,{"DimIds":[40,1],"IsNull":false,"Value":0.8,"SubId":0}
]}
```

Example 2:

JSON body:

```
{
  "Name":       "ageSex",
  "Offset":     6,
  "Size":       4,
  "IsLastPage": true
}
```

Result:

{"Layout":{"Offset":6,"Size":2,"IsLastPage":true}
,"Page":[{"DimIds":[40,0],"IsNull":false,"Value":0.7,"SubId":0}
,{"DimIds":[40,1],"IsNull":false,"Value":0.8,"SubId":0}
]}

Example 3:

JSON body:

```
{
  "Name": "ageSex",
  "Offset": 2,
  "OrderBy": [{
      "IndexOne": 2,
      "IsDesc": true
    }, {
      "IndexOne": 3,
      "IsDesc": true
    }
  ]
}
```

Result:

{"Layout":{"Offset":2,"Size":6,"IsLastPage":true}
,"Page":[{"DimIds":[30,1],"IsNull":false,"Value":0.6,"SubId":0}
,{"DimIds":[30,0],"IsNull":false,"Value":0.5,"SubId":0}
,{"DimIds":[20,1],"IsNull":false,"Value":0.4,"SubId":0}
,{"DimIds":[20,0],"IsNull":false,"Value":0.3,"SubId":0}
,{"DimIds":[10,1],"IsNull":false,"Value":0.2,"SubId":0}
,{"DimIds":[10,0],"IsNull":false,"Value":0.1,"SubId":0}
]}

Example 4:

JSON body:

```
{
  "Name": "ageSex",
  "Offset": 0,
  "Size": 100,
  "FilterById": [{
      "Name": "dim0",
      "Op": "IN",
      "EnumIds": [20, 40]
    }, {
      "Name": "dim1",
      "Op": "=",
      "EnumIds": [1]
    }
  ],
  "OrderBy": [{
      "IndexOne": 2,
      "IsDesc": true
    }, {
      "IndexOne": 3,
      "IsDesc": true
    }
  ]
}
```

Result:

{"Layout":{"Offset":0,"Size":2,"IsLastPage":true}
,"Page":[{"DimIds":[40,1],"IsNull":false,"Value":0.8,"SubId":0}
,{"DimIds":[20,1],"IsNull":false,"Value":0.4,"SubId":0}
]}

Read output table values from model run

Read a "page" of output table values from model run.

- Page is part of output table values defined by zero-based "start" row number and row count. If row count <= 0 then all rows below start row number returned.
- Dimension(s) and enum-based parameters returned as enum codes. If dimension type or parameter type is simple (integer or boolean) then string value used (ex.: "true", "1234").
- Values can be from output table **expressions, accumulators or derived accumulators**.
- Method verb must be POST and Content-Type header "application/json".

JSON body POSTed to specify output table name, page size, row count, filters and row order. It is expected to be JSON representation of db.ReadLayout structure from Go library.

Method:

```
POST /api/model/:model/run/:run/table/value
```

For example:

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/run/Default/table/value -d @test.json
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/run/2019_01_17_19_59_52_998/table/value -d @test.json
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

JSON body arguments:

For example:

```
{
  "Name": "salarySex",
  "Offset": 0,
  "Size": 100,
  "IsLastPage": true,
  "Filter": [{
      "Name": "dim0",
      "Op": "IN",
      "Values": ["L", "H"]
    }, {
      "Name": "dim1",
      "Op": "BETWEEN",
      "Values": ["F", "all"]
    }
  ],
  "OrderBy": [{
      "IndexOne": 2,
      "IsDesc": true
    }, {
      "IndexOne": 3,
      "IsDesc": true
    }
  ],
  "ValueName": "acc2",
  "IsAccum": true,
  "IsAllAccum": true
}
```

```
Name       - (required) output table name
Offset     - (optional) zero-based start row to select output table values
Size       - (optional) max row count to select output table values.
IsLastPage - (optional) if true then always return non-empty last page of data.
Filter     - (optional) conditions to filter dimension enum id's
OrderBy    - (optional) list of columns indexes (one based) to order by
ValueName  - (optional) if not empty then only that value selected (ex.: "acc2"), default: all values
IsAccum    - (optional) if true then select accumulators
IsAllAccum - (optional) if true then select from "all accumulators" view else from accumulators table
```

By default oms service selects 100 rows (it can be configured). If `Size` <= 0 specified then all rows selected.

Filter conditions joined by AND and can have following operations:

```
=       - enum equal to:          AgeGroup = "20-30"
!=      - enum not equal to:      AgeGroup <> "20-30"
>       - enum greater than:      AgeGroup > "20-30"
>=      - enum greater or equal:  AgeGroup >= "20-30"
<       - enum less than:         AgeGroup < "20-30"
<=      - enum less or equal:     AgeGroup <= "20-30"
IN      - enum is in the list of: AgeGroup IN ("20-30", "30-40", "40+")
BETWEEN - between min and max:    AgeGroup BETWEEN "30-40" AND "all"
IN_AUTO - automatically choose most suitable: = or != or IN or BETWEEN
```

Keep in mind: dimension enums are always ordered by id's, not by code and result of filter `Sex < "M"` may not be `Sex = "F"`.

Order by specified by one-based column(s) index(es) in result. Columns always contain enum id's, not enum codes and therefore result ordered by id's

In case of output table expressions columns are:

```
SELECT expr_id, dim0, dim1, ..., expr_value FROM valueTable ORDER BY 1, 2,...
```

In case of output table accumulators columns are:

```
SELECT acc_id, sub_id, dim0, dim1, ..., acc_value FROM accumulatorTable ORDER BY 1, 2,...
```

In case of "all accumulators" columns are:

```
SELECT sub_id, dim0, dim1, ..., acc0, acc1,... FROM allAccumulatorsView ORDER BY 1, 2,...
```

JSON response:

```
{
  Layout: {
    Offset:     actual first row number of the page data (zero-base),
    Size:       actual data page row count,
    IsLastPage: true if this is last page of data
  },
  Page: [....page of data...]
}
```

Example 1:

JSON body:

```
{
  "Name": "salarySex",
  "Filter": [],
  "OrderBy": []
}
```

Result:

```
< HTTP/1.1 200 OK
< Access-Control-Allow-Origin: *
< Content-Type: application/json
< Date: Tue, 19 Dec 2017 18:43:54 GMT
< Transfer-Encoding: chunked
<
{"Layout":{"Offset":0,"Size":36,"IsLastPage":true}
,"Page":[{"Dims":["L","M"],"Value":50,"IsNull":false,"ExprId":0}
,{"Dims":["L","F"],"Value":60,"IsNull":false,"ExprId":0}
,{"Dims":["L","all"],"Value":1,"IsNull":false,"ExprId":0}
,{"Dims":["M","M"],"Value":51.599999999999994,"IsNull":false,"ExprId":0}
,{"Dims":["M","F"],"Value":62,"IsNull":false,"ExprId":0}
,{"Dims":["M","all"],"Value":2,"IsNull":false,"ExprId":0}
,{"Dims":["H","M"],"Value":53.2,"IsNull":false,"ExprId":0}
,{"Dims":["H","F"],"Value":64,"IsNull":false,"ExprId":0}
,{"Dims":["H","all"],"Value":3,"IsNull":false,"ExprId":0}
,{"Dims":["L","M"],"Value":1,"IsNull":false,"ExprId":1}
,{"Dims":["L","F"],"Value":2,"IsNull":false,"ExprId":1}
,{"Dims":["L","all"],"Value":801,"IsNull":false,"ExprId":1}
,{"Dims":["M","M"],"Value":3,"IsNull":false,"ExprId":1}
,{"Dims":["M","F"],"Value":4,"IsNull":false,"ExprId":1}
,{"Dims":["M","all"],"Value":803,"IsNull":false,"ExprId":1}
,{"Dims":["H","M"],"Value":4,"IsNull":false,"ExprId":1}
,{"Dims":["H","F"],"Value":5,"IsNull":false,"ExprId":1}
,{"Dims":["H","all"],"Value":804,"IsNull":false,"ExprId":1}
,{"Dims":["L","M"],"Value":50,"IsNull":false,"ExprId":2}
,{"Dims":["L","F"],"Value":60,"IsNull":false,"ExprId":2}
,{"Dims":["L","all"],"Value":1,"IsNull":false,"ExprId":2}
,{"Dims":["M","M"],"Value":51.599999999999994,"IsNull":false,"ExprId":2}
,{"Dims":["M","F"],"Value":62,"IsNull":false,"ExprId":2}
,{"Dims":["M","all"],"Value":2,"IsNull":false,"ExprId":2}
,{"Dims":["H","M"],"Value":53.2,"IsNull":false,"ExprId":2}
,{"Dims":["H","F"],"Value":64,"IsNull":false,"ExprId":2}
,{"Dims":["H","all"],"Value":3,"IsNull":false,"ExprId":2}
,{"Dims":["L","M"],"Value":50,"IsNull":false,"ExprId":3}
,{"Dims":["L","F"],"Value":120,"IsNull":false,"ExprId":3}
,{"Dims":["L","all"],"Value":801,"IsNull":false,"ExprId":3}
,{"Dims":["M","M"],"Value":154.79999999999998,"IsNull":false,"ExprId":3}
,{"Dims":["M","F"],"Value":248,"IsNull":false,"ExprId":3}
,{"Dims":["M","all"],"Value":1606,"IsNull":false,"ExprId":3}
,{"Dims":["H","M"],"Value":212.8,"IsNull":false,"ExprId":3}
,{"Dims":["H","F"],"Value":320,"IsNull":false,"ExprId":3}
,{"Dims":["H","all"],"Value":2412,"IsNull":false,"ExprId":3}
]}
```

Example 2:

JSON body:

```
{
  "Name":       "salarySex",
  "Offset":     32,
  "Size":       8,
  "IsLastPage": true,
  "Filter":     [],
  "OrderBy":    []
}
```

Result:


```
{"Layout":{"Offset":32,"Size":4,"IsLastPage":true}
,"Page":[{"Dims":["M","all"],"Value":1606,"IsNull":false,"ExprId":3}
,{"Dims":["H","M"],"Value":212.8,"IsNull":false,"ExprId":3}
,{"Dims":["H","F"],"Value":320,"IsNull":false,"ExprId":3}
,{"Dims":["H","all"],"Value":2412,"IsNull":false,"ExprId":3}
]}
```


Example 3:

JSON body:

```
{
  "Name": "salarySex",
  "Filter": [],
  "OrderBy": [{
      "IndexOne": 2,
      "IsDesc": true
    }, {
      "IndexOne": 3,
      "IsDesc": true
    }
  ],
  "IsAccum": true,
  "IsAllAccum": false
}
```

Result:


```
{"Layout":{"Offset":0,"Size":18,"IsLastPage":true}
,"Page":[{"Dims":["H","M"],"Value":53.2,"IsNull":false,"AccId":0,"SubId":0}
,{"Dims":["H","F"],"Value":64,"IsNull":false,"AccId":0,"SubId":0}
,{"Dims":["H","all"],"Value":3,"IsNull":false,"AccId":0,"SubId":0}
,{"Dims":["H","M"],"Value":4,"IsNull":false,"AccId":1,"SubId":0}
,{"Dims":["H","F"],"Value":5,"IsNull":false,"AccId":1,"SubId":0}
,{"Dims":["H","all"],"Value":804,"IsNull":false,"AccId":1,"SubId":0}
,{"Dims":["M","M"],"Value":51.599999999999994,"IsNull":false,"AccId":0,"SubId":0}
,{"Dims":["M","F"],"Value":62,"IsNull":false,"AccId":0,"SubId":0}
,{"Dims":["M","all"],"Value":2,"IsNull":false,"AccId":0,"SubId":0}
,{"Dims":["M","M"],"Value":3,"IsNull":false,"AccId":1,"SubId":0}
,{"Dims":["M","F"],"Value":4,"IsNull":false,"AccId":1,"SubId":0}
,{"Dims":["M","all"],"Value":803,"IsNull":false,"AccId":1,"SubId":0}
,{"Dims":["L","M"],"Value":50,"IsNull":false,"AccId":0,"SubId":0}
,{"Dims":["L","F"],"Value":60,"IsNull":false,"AccId":0,"SubId":0}
,{"Dims":["L","all"],"Value":1,"IsNull":false,"AccId":0,"SubId":0}
,{"Dims":["L","M"],"Value":1,"IsNull":false,"AccId":1,"SubId":0}
,{"Dims":["L","F"],"Value":2,"IsNull":false,"AccId":1,"SubId":0}
,{"Dims":["L","all"],"Value":801,"IsNull":false,"AccId":1,"SubId":0}
]}
```


Example 4:

JSON body:

```
{
  "Name": "salarySex",
  "Offset": 0,
  "Size": 100,
  "Filter": [{
      "Name": "dim0",
      "Op": "IN",
      "Values": ["L", "H"]
    }, {
      "Name": "dim1",
      "Op": "BETWEEN",
      "Values": ["F", "all"]
    }
  ],
  "OrderBy": [{
      "IndexOne": 2,
      "IsDesc": true
    }, {
      "IndexOne": 3,
      "IsDesc": true
    }
  ],
  "ValueName": "acc2",
  "IsAccum": true,
  "IsAllAccum": true
}
```

Result:


```
{"Layout":{"Offset":0,"Size":4,"IsLastPage":true}
,"Page":[{"Dims":["H","all"],"SubId":0,"IsNull":[false],"Value":[807]}
,{"Dims":["H","F"],"SubId":0,"IsNull":[false],"Value":[69]}
,{"Dims":["L","all"],"SubId":0,"IsNull":[false],"Value":[802]}
,{"Dims":["L","F"],"SubId":0,"IsNull":[false],"Value":[62]}
]}
```

Read output table values from model run (enum id's)

Read a "page" of output table values from model run.

- Page is part of output table values defined by zero-based "start" row number and row count. If row count <= 0 then all rows below start row number returned.
- Dimension(s) returned as enum id, not enum codes.
- Values can be from output table **expressions, accumulators or derived accumulators**.
- Method verb must be POST and Content-Type header "application/json".

JSON body POSTed to specify output table name, page size, row count, filters and row order. It is expected to be JSON representation of db.ReadLayout structure from Go library.

Method:

```
POST /api/model/:model/run/:run/table/value-id
```

For example:

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/run/Default/table/value-id -d @test.json
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/run/2019_01_17_19_59_52_998/table/value-id -d @test.json
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

JSON body arguments:

For example:

```
{
  "Name": "salarySex",
  "Offset": 0,
  "Size": 100,
  "IsLastPage": true,
  "FilterById": [{
      "Name": "AgeGroup",
      "Op": "IN",
      "EnumIds": [100, 300]
    }, {
      "Name": "Province",
      "Op": "BETWEEN",
      "EnumIds": [1, 800]
    }
  ],
  "OrderBy": [{
      "IndexOne": 2,
      "IsDesc": true
    }, {
      "IndexOne": 3,
      "IsDesc": true
    }
  ],
  "ValueName": "acc2",
  "IsAccum": true,
  "IsAllAccum": true
}
```

```
Name       - (required) output table name
Offset     - (optional) zero-based start row to select output table values
Size       - (optional) max row count to select output table values.
IsLastPage - (optional) if true then always return non-empty last page of data.
FilterById - (optional) conditions to filter dimension enum id's
OrderBy    - (optional) list of columns indexes (one based) to order by
ValueName  - (optional) if not empty then only that value selected (ex.: "acc2"), default: all values
IsAccum    - (optional) if true then select accumulators
IsAllAccum - (optional) if true then select from "all accumulators" view else from accumulators table
```

By default oms service selects 100 rows (it can be configured). If `Size` <= 0 specified then all rows selected.

Filter conditions joined by AND and can have following operations:

```
=       - enum id equal to:         AgeGroup = 20
!=      - enum id not equal to:     AgeGroup <> 20
>       - enum id greater than:     AgeGroup > 20
>=      - enum id greater or equal: AgeGroup >= 20
<       - enum id less than:        AgeGroup < 20
<=      - enum id less or equal:    AgeGroup <= 20
IN      - in the list of id's:      AgeGroup IN (20, 30, 40)
BETWEEN - between min and max:      AgeGroup BETWEEN 20 AND 40
IN_AUTO - automatically choose most suitable: = or != or IN or BETWEEN
```

Order by specified by one-based column(s) index(es) in result. Columns always contain enum id's, not enum codes and therefore result ordered by id's

In case of output table expressions columns are:

```
SELECT expr_id, dim0, dim1, ..., expr_value FROM valueTable ORDER BY 1, 2,...
```

In case of output table accumulators columns are:

```
SELECT acc_id, sub_id, dim0, dim1, ..., acc_value FROM accumulatorTable ORDER BY 1, 2,...
```

In case of "all accumulators" columns are:

```
SELECT sub_id, dim0, dim1, ..., acc0, acc1,... FROM allAccumulatorsView ORDER BY 1, 2,...
```

Example 1:

JSON body:

```
{
  "Name": "salarySex"
}
```

Result:


```
< Access-Control-Allow-Origin: *
< Content-Type: application/json
< Date: Tue, 19 Dec 2017 19:04:15 GMT
< Transfer-Encoding: chunked
<
{"Layout":{"Offset":0,"Size":36,"IsLastPage":true}
,"Page":[{"DimIds":[100,0],"Value":50,"IsNull":false,"ExprId":0}
,{"DimIds":[100,1],"Value":60,"IsNull":false,"ExprId":0}
,{"DimIds":[100,800],"Value":1,"IsNull":false,"ExprId":0}
,{"DimIds":[200,0],"Value":51.599999999999994,"IsNull":false,"ExprId":0}
,{"DimIds":[200,1],"Value":62,"IsNull":false,"ExprId":0}
,{"DimIds":[200,800],"Value":2,"IsNull":false,"ExprId":0}
,{"DimIds":[300,0],"Value":53.2,"IsNull":false,"ExprId":0}
,{"DimIds":[300,1],"Value":64,"IsNull":false,"ExprId":0}
,{"DimIds":[300,800],"Value":3,"IsNull":false,"ExprId":0}
,{"DimIds":[100,0],"Value":1,"IsNull":false,"ExprId":1}
,{"DimIds":[100,1],"Value":2,"IsNull":false,"ExprId":1}
,{"DimIds":[100,800],"Value":801,"IsNull":false,"ExprId":1}
,{"DimIds":[200,0],"Value":3,"IsNull":false,"ExprId":1}
,{"DimIds":[200,1],"Value":4,"IsNull":false,"ExprId":1}
,{"DimIds":[200,800],"Value":803,"IsNull":false,"ExprId":1}
,{"DimIds":[300,0],"Value":4,"IsNull":false,"ExprId":1}
,{"DimIds":[300,1],"Value":5,"IsNull":false,"ExprId":1}
,{"DimIds":[300,800],"Value":804,"IsNull":false,"ExprId":1}
,{"DimIds":[100,0],"Value":50,"IsNull":false,"ExprId":2}
,{"DimIds":[100,1],"Value":60,"IsNull":false,"ExprId":2}
,{"DimIds":[100,800],"Value":1,"IsNull":false,"ExprId":2}
,{"DimIds":[200,0],"Value":51.599999999999994,"IsNull":false,"ExprId":2}
,{"DimIds":[200,1],"Value":62,"IsNull":false,"ExprId":2}
,{"DimIds":[200,800],"Value":2,"IsNull":false,"ExprId":2}
,{"DimIds":[300,0],"Value":53.2,"IsNull":false,"ExprId":2}
,{"DimIds":[300,1],"Value":64,"IsNull":false,"ExprId":2}
,{"DimIds":[300,800],"Value":3,"IsNull":false,"ExprId":2}
,{"DimIds":[100,0],"Value":50,"IsNull":false,"ExprId":3}
,{"DimIds":[100,1],"Value":120,"IsNull":false,"ExprId":3}
,{"DimIds":[100,800],"Value":801,"IsNull":false,"ExprId":3}
,{"DimIds":[200,0],"Value":154.79999999999998,"IsNull":false,"ExprId":3}
,{"DimIds":[200,1],"Value":248,"IsNull":false,"ExprId":3}
,{"DimIds":[200,800],"Value":1606,"IsNull":false,"ExprId":3}
,{"DimIds":[300,0],"Value":212.8,"IsNull":false,"ExprId":3}
,{"DimIds":[300,1],"Value":320,"IsNull":false,"ExprId":3}
,{"DimIds":[300,800],"Value":2412,"IsNull":false,"ExprId":3}
]}
```


Example 2:

JSON body:

```
{
  "Name":       "salarySex",
  "Offset":     32,
  "Size":       8,
  "IsLastPage": true
}
```

Result:


```
{"Layout":{"Offset":32,"Size":4,"IsLastPage":true}
,"Page":[{"DimIds":[200,800],"Value":1606,"IsNull":false,"ExprId":3}
,{"DimIds":[300,0],"Value":212.8,"IsNull":false,"ExprId":3}
,{"DimIds":[300,1],"Value":320,"IsNull":false,"ExprId":3}
,{"DimIds":[300,800],"Value":2412,"IsNull":false,"ExprId":3}
]}
```


Example 3:

JSON body:

```
{
  "Name": "salarySex",
  "FilterById": [],
  "OrderBy": [{
      "IndexOne": 2,
      "IsDesc": true
    }, {
      "IndexOne": 3,
      "IsDesc": true
    }
  ],
  "IsAccum": true,
  "IsAllAccum": false
}
```

Result:

```
{"Layout":{"Offset":0,"Size":18,"IsLastPage":true}
,"Page":[{"DimIds":[300,0],"Value":53.2,"IsNull":false,"AccId":0,"SubId":0}
,{"DimIds":[300,1],"Value":64,"IsNull":false,"AccId":0,"SubId":0}
,{"DimIds":[300,800],"Value":3,"IsNull":false,"AccId":0,"SubId":0}
,{"DimIds":[300,0],"Value":4,"IsNull":false,"AccId":1,"SubId":0}
,{"DimIds":[300,1],"Value":5,"IsNull":false,"AccId":1,"SubId":0}
,{"DimIds":[300,800],"Value":804,"IsNull":false,"AccId":1,"SubId":0}
,{"DimIds":[200,0],"Value":51.599999999999994,"IsNull":false,"AccId":0,"SubId":0}
,{"DimIds":[200,1],"Value":62,"IsNull":false,"AccId":0,"SubId":0}
,{"DimIds":[200,800],"Value":2,"IsNull":false,"AccId":0,"SubId":0}
,{"DimIds":[200,0],"Value":3,"IsNull":false,"AccId":1,"SubId":0}
,{"DimIds":[200,1],"Value":4,"IsNull":false,"AccId":1,"SubId":0}
,{"DimIds":[200,800],"Value":803,"IsNull":false,"AccId":1,"SubId":0}
,{"DimIds":[100,0],"Value":50,"IsNull":false,"AccId":0,"SubId":0}
,{"DimIds":[100,1],"Value":60,"IsNull":false,"AccId":0,"SubId":0}
,{"DimIds":[100,800],"Value":1,"IsNull":false,"AccId":0,"SubId":0}
,{"DimIds":[100,0],"Value":1,"IsNull":false,"AccId":1,"SubId":0}
,{"DimIds":[100,1],"Value":2,"IsNull":false,"AccId":1,"SubId":0}
,{"DimIds":[100,800],"Value":801,"IsNull":false,"AccId":1,"SubId":0}
]}
```

Example 3:

JSON body:

```
{
  "Name": "salarySex",
  "Offset": 0,
  "Size": 100,
  "FilterById": [{
      "Name": "dim0",
      "Op": "IN",
      "EnumIds": [100, 300]
    }, {
      "Name": "dim1",
      "Op": "BETWEEN",
      "EnumIds": [1, 800]
    }
  ],
  "OrderBy": [{
      "IndexOne": 2,
      "IsDesc": true
    }, {
      "IndexOne": 3,
      "IsDesc": true
    }
  ],
  "ValueName": "acc2",
  "IsAccum": true,
  "IsAllAccum": true
}
```

Result:

```
{"Layout":{"Offset":0,"Size":4,"IsLastPage":true}
,"Page":[{"DimIds":[300,800],"SubId":0,"IsNull":[false],"Value":[807]}
,{"DimIds":[300,1],"SubId":0,"IsNull":[false],"Value":[69]}
,{"DimIds":[100,800],"SubId":0,"IsNull":[false],"Value":[802]}
,{"DimIds":[100,1],"SubId":0,"IsNull":[false],"Value":[62]}
]}
```

Read microdata values from model run

Read a "page" of microdata values from model run.

Page is part of microdata values defined by zero-based "start" row number and row count. If row count <= 0 then all rows below start row number returned.

Enum-based microdata attributes returned as enum codes.

Method verb must be POST and Content-Type header "application/json". JSON body POSTed to specify microdata name, page size, row count, filters and row order. It is expected to be JSON representation of db.ReadLayout structure from Go library.

Method:

```
POST /api/model/:model/run/:run/microdata/value
```

For example:

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/value -d @test.json
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/run/2016_08_17_21_07_55_123/microdata/value -d @test.json
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

JSON body arguments:

For example:

```
{
  "Name": "Person",
  "Offset": 8,
  "Size": 16,
  "Filter": [{
      "Name": "Age",
      "Op": "BETWEEN",
      "Values": ["21", "65"]
    }, {
      "Name": "Province",
      "Op": "IN",
      "Values": ["BC", "QC"]
    }, {
      "Name": "Sex",
      "Op": "=",
      "Values": ["F"]
    }
  ],
  "OrderBy": [{
      "IndexOne": 5
    }, {
      "IndexOne": 6,
      "IsDesc": true
    }
  ]
}
```

```
Name       - (required) microdata name
Offset     - (optional) zero-based start row to select microdata values
Size       - (optional) max row count to select microdata values.
IsLastPage - (optional) if true then always return non-empty last page of data.
Filter     - (optional) conditions to filter attribute values, if attribute is enum-based then filter by enum code(s).
OrderBy    - (optional) list of columns indexes (one based) to order by
```

By default oms service selects 100 rows (it can be configured). If `Size` <= 0 specified then all rows selected.

Filter conditions joined by AND and can have following operations:

```
=       - value equal to:          Age = 20
!=      - value not equal to:      Age <> 20
>       - value greater than:      Age > 20
>=      - value greater or equal:  Age >= 20
<       - value less than:         Age < 20
<=      - value less or equal:     Age <= 20
IN      - in the list of codes:    Province IN ("BC", "QC", "ON")
BETWEEN - between min and max:     Age BETWEEN 20 AND 40
IN_AUTO - automatically choose most suitable: = or != or IN or BETWEEN
```

Order by specified by one-based column(s) index(es) in result. In case of microdata columns are:

```
 SELECT entity_key, attr0, attr1, ..., value FROM microdataTable ORDER BY 1, 2,...
```

For enum-based attribute column always contain enum id's, not enum codes and therefore result ordered by id's

JSON response:

```
{
  Layout: {
    Offset:     actual first row number of the page data (zero-base),
    Size:       actual data page row count,
    IsLastPage: true if this is last page of data
  },
  Page: [....page of data...]
}
```

Example 1:

JSON body:

```
{
  "Name": "Person",
  "Offset": 8,
  "Size": 2
}
```

Result:

```
< HTTP/1.1 200 OK
< Content-Type: application/json
< Date: Fri, 13 Jan 2023 22:25:27 GMT
< Content-Length: 1299
<
{
    "Page": [{
            "Key": 8,
            "Attr": [{
                    "IsNull": false,
                    "Value": "39"
                }, {
                    "IsNull": false,
                    "Value": "30-40"
                }, {
                    "IsNull": false,
                    "Value": "237539"
                }
            ]
        }, {
            "Key": 9,
            "Attr": [{
                    "IsNull": false,
                    "Value": "30"
                }, {
                    "IsNull": false,
                    "Value": "30-40"
                }, {
                    "IsNull": false,
                    "Value": "245730"
                }
            ]
        }
    ],
    "Layout": {
        "Offset": 8,
        "Size": 2,
        "IsLastPage": false
    }
}
```

Example 2:

JSON body:

```
{
  "Name": "Person",
  "Offset": 0,
  "Size": 3,
  "Filter": [{
      "Name": "AgeGroup",
      "Op": "IN",
      "Values": ["20-30", "40+"]
    }, {
      "Name": "Sex",
      "Op": "=",
      "Values": ["F"]
    }
  ],
  "OrderBy": [{
      "IndexOne": 5
    }, {
      "IndexOne": 6,
      "IsDesc": true
    }
  ]
}
```

Result:

```
< HTTP/1.1 200 OK
< Content-Type: application/json
< Date: Fri, 13 Jan 2023 22:49:06 GMT
< Content-Length: 1000
<
{
    "Page": [{
            "Key": 844424930131977,
            "Attr": [{
                    "IsNull": false,
                    "Value": "83"
                }, {
```

```
                "IsNull": false,
                "Value": "40+"
            }, {
                "IsNull": false,
                "Value": "F"
            }, {
                "IsNull": false,
                "Value": "23000"
            }, {
                "IsNull": false,
                "Value": "0"
            }, {
                "IsNull": false,
                "Value": "L"
            }, {
                "IsNull": false,
                "Value": "Part"
            }, {
                "IsNull": false,
                "Value": "true"
            }, {
                "IsNull": false,
                "Value": "23000"
            }
        ]
    }, {
        "Key": 562949953421322,
        "Attr": [{
                "IsNull": false,
                "Value": "83"
            }, {
                "IsNull": false,
                "Value": "40+"
            }, {
                "IsNull": false,
                "Value": "F"
            }, {
                "IsNull": false,
                "Value": "23000"
            }, {
                "IsNull": false,
                "Value": "0"
            }, {
                "IsNull": false,
                "Value": "L"
            }, {
                "IsNull": false,
                "Value": "Part"
            }, {
                "IsNull": false,
                "Value": "true"
            }, {
                "IsNull": false,
                "Value": "23000"
            }
        ]
    }, {
        "Key": 281474976710667,
        "Attr": [{
                "IsNull": false,
                "Value": "83"
            }, {
                "IsNull": false,
                "Value": "40+"
            }, {
                "IsNull": false,
                "Value": "F"
            }, {
                "IsNull": false,
                "Value": "23000"
            }, {
                "IsNull": false,
                "Value": "0"
            }, {
                "IsNull": false,
                "Value": "L"
            }, {
                "IsNull": false,
                "Value": "Part"
            }, {
                "IsNull": false,
                "Value": "true"
            }, {
                "IsNull": false,
                "Value": "23000"
            }
        ]
    }
```

```
    ],
    "Layout": {
        "Offset": 0,
        "Size": 3,
        "IsLastPage": false
    }
}
```

Read microdata values from model run (enum id's)

Read a "page" of microdata values from model run.

Page is part of microdata values defined by zero-based "start" row number and row count. If row count <= 0 then all rows below start row number returned.

Enum-based microdata attributes returned as enum id, not enum codes.

Method verb must be POST and Content-Type header "application/json". JSON body POSTed to specify microdata name, page size, row count, filters and row order. It is expected to be JSON representation of db.ReadLayout structure from Go library.

Method:

```
POST /api/model/:model/run/:run/microdata/value-id
```

For example:

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/value-id -d @test.json
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/run/2016_08_17_21_07_55_123/microdata/value-id -d @test.json
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

JSON body arguments:

For example:

```
{
  "Name": "Person",
  "Offset": 8,
  "Size": 16,
  "FilterById": [{
      "Name": "AgeGroup",
      "Op": "IN",
      "EnumIds": [20, 40]
    }, {
      "Name": "Sex",
      "Op": "=",
      "EnumIds": [1]
    }
  ],
  "OrderBy": [{
      "IndexOne": 5
    }, {
      "IndexOne": 6,
      "IsDesc": true
    }
  ]
}
```

```
Name       - (required) microdata name
Offset     - (optional) zero-based start row to select microdata values
Size       - (optional) max row count to select microdata values.
IsLastPage - (optional) if true then always return non-empty last page of data.
FilterById - (optional) conditions to filter  attribute values, if attribute is enum-based then filter by enum id(s).
OrderBy    - (optional) list of columns indexes (one based) to order by
```

By default oms service selects 100 rows (it can be configured). If `Size` <= 0 specified then all rows selected.

Filter conditions joined by AND and can have following operations:

```
=       - value equal to:         Age = 20
!=      - value not equal to:     Age <> 20
>       - value greater than:     Age > 20
>=      - value greater or equal: Age >= 20
<       - value less than:        Age < 20
<=      - value less or equal:    Age <= 20
IN      - in the list of id's:    Age IN (20, 30, 40)
BETWEEN - between min and max:    Age BETWEEN 20 AND 40
IN_AUTO - automatically choose most suitable: = or != or IN or BETWEEN
```

Order by specified by one-based column(s) index(es) in result. In case of microdata columns are:

```
SELECT entity_key, attr0, attr1, ..., value FROM microdataTable ORDER BY 1, 2,...
```

For enum-based attribute column always contain enum id's, not enum codes and therefore result ordered by id's

JSON response:

```
{
  Layout: {
    Offset:     actual first row number of the page data (zero-base),
    Size:       actual data page row count,
    IsLastPage: true if this is last page of data
  },
  Page: [....page of data...]
}
```

Example 1:

JSON body:

```
{
  "Name": "Other",
  "Offset": 8,
  "Size": 2
}
```

Result:

```
< HTTP/1.1 200 OK
< Content-Type: application/json
< Date: Fri, 13 Jan 2023 22:41:11 GMT
< Content-Length: 279
<
{
    "Page": [{
            "Key": 8,
            "Attr": [{
                    "IsNull": false,
                    "Value": 39
                }, {
                    "IsNull": false,
                    "Value": 30
                }, {
                    "IsNull": false,
                    "Value": 237539
                }
            ]
        }, {
            "Key": 9,
            "Attr": [{
                    "IsNull": false,
                    "Value": 30
                }, {
                    "IsNull": false,
                    "Value": 30
                }, {
                    "IsNull": false,
                    "Value": 245730
                }
            ]
        }
    ],
    "Layout": {
        "Offset": 8,
        "Size": 2,
        "IsLastPage": false
    }
}
```

Example 2:

JSON body:

```
{
  "Name": "Person",
  "Offset": 0,
  "Size": 3,
  "FilterById": [{
      "Name": "AgeGroup",
      "Op": "IN",
      "EnumIds": [20, 40]
    }, {
      "Name": "Sex",
      "Op": "=",
      "EnumIds": [1]
    }
  ],
  "OrderBy": [{
      "IndexOne": 5
    }, {
      "IndexOne": 6,
      "IsDesc": true
    }
  ]
}
```

Result:

```
< HTTP/1.1 200 OK
< Content-Type: application/json
< Date: Fri, 13 Jan 2023 22:52:12 GMT
< Content-Length: 943
<
{
    "Page": [{
            "Key": 844424930131977,
            "Attr": [{
                    "IsNull": false,
                    "Value": 83
                }, {
```

```
            "IsNull": false,
            "Value": 40
        }, {
            "IsNull": false,
            "Value": 1
        }, {
            "IsNull": false,
            "Value": 23000
        }, {
            "IsNull": false,
            "Value": 0
        }, {
            "IsNull": false,
            "Value": 100
        }, {
            "IsNull": false,
            "Value": 33
        }, {
            "IsNull": false,
            "Value": true
        }, {
            "IsNull": false,
            "Value": 23000
        }
    ]
}, {
    "Key": 562949953421322,
    "Attr": [{
            "IsNull": false,
            "Value": 83
        }, {
            "IsNull": false,
            "Value": 40
        }, {
            "IsNull": false,
            "Value": 1
        }, {
            "IsNull": false,
            "Value": 23000
        }, {
            "IsNull": false,
            "Value": 0
        }, {
            "IsNull": false,
            "Value": 100
        }, {
            "IsNull": false,
            "Value": 33
        }, {
            "IsNull": false,
            "Value": true
        }, {
            "IsNull": false,
            "Value": 23000
        }
    ]
}, {
    "Key": 281474976710667,
    "Attr": [{
            "IsNull": false,
            "Value": 83
        }, {
            "IsNull": false,
            "Value": 40
        }, {
            "IsNull": false,
            "Value": 1
        }, {
            "IsNull": false,
            "Value": 23000
        }, {
            "IsNull": false,
            "Value": 0
        }, {
            "IsNull": false,
            "Value": 100
        }, {
            "IsNull": false,
            "Value": 33
        }, {
            "IsNull": false,
            "Value": true
        }, {
            "IsNull": false,
            "Value": 23000
        }
    ]
}
```

```
        }
    ],
    "Layout": {
        "Offset": 0,
        "Size": 3,
        "IsLastPage": false
    }
}
```

GET parameter values from workset

Read a "page" of parameter values from workset.

Page is part of parameter values defined by zero-based "start" row number and row count. If row count <= 0 then all rows returned.

Dimension(s) and enum-based parameters returned as enum codes.

Methods:

```
GET /api/model/:model/workset/:set/parameter/:name/value
GET /api/model/:model/workset/:set/parameter/:name/value/start/:start
GET /api/model/:model/workset/:set/parameter/:name/value/start/:start/count/:count
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:set - (required) workset name
```

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default".

```
:name - (required) parameter name
```

```
:start - (optional) start "page" row number, zero-based.
:count - (optional) "page" size, number of rows to select.
```

By default oms service selects 100 rows (it can be configured). If count <= 0 specified then all rows selected.

Call examples:

```
http://localhost:4040/api/model/modelOne/workset/modelOne_other/parameter/ageSex/value
http://localhost:4040/api/model/modelOne/workset/modelOne_other/parameter/ageSex/value/start/2
http://localhost:4040/api/model/modelOne/workset/modelOne_other/parameter/ageSex/value/start/2/count/4
http://localhost:4040/api/model/_201208171604590148_/workset/modelOne_set/parameter/ageSex/value
```

Return example:


```
[{"Dims":["10-20","M"],"IsNull":false,"Value":1.1,"SubId":0}
,{"Dims":["10-20","F"],"IsNull":false,"Value":1.2,"SubId":0}
,{"Dims":["20-30","M"],"IsNull":false,"Value":1.3,"SubId":0}
,{"Dims":["20-30","F"],"IsNull":false,"Value":1.4,"SubId":0}
,{"Dims":["30-40","M"],"IsNull":false,"Value":1.5,"SubId":0}
,{"Dims":["30-40","F"],"IsNull":false,"Value":1.6,"SubId":0}
,{"Dims":["40+","M"],"IsNull":false,"Value":1.7,"SubId":0}
,{"Dims":["40+","F"],"IsNull":false,"Value":1.8,"SubId":0}
]
```

GET parameter values from model run

Read a "page" of parameter values from model run.

Page is part of parameter values defined by zero-based "start" row number and row count. If row count <= 0 then all rows returned.

Dimension(s) and enum-based parameters returned as enum codes.

Methods:

```
GET /api/model/:model/run/:run/parameter/:name/value
GET /api/model/:model/run/:run/parameter/:name/value/start/:start
GET /api/model/:model/run/:run/parameter/:name/value/start/:start/count/:count
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

```
:name - (required) parameter name
```

```
:start - (optional) start "page" row number, zero-based.
:count - (optional) "page" size, number of rows to select.
```

By default oms service selects 100 rows (it can be configured). If count <= 0 specified then all rows selected.

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Default/parameter/ageSex/value
http://localhost:4040/api/model/modelOne/run/Default/parameter/ageSex/value/start/2
http://localhost:4040/api/model/modelOne/run/Default/parameter/ageSex/value/start/2/count/4
http://localhost:4040/api/model/_201208171604590148_/run/f172e98da17beb058f30f11768053456/parameter/ageSex/value
http://localhost:4040/api/model/_201208171604590148_/run/2019_01_17_19_59_52_998/parameter/ageSex/value
```

Return example:


```
[{"Dims":["10-20","M"],"IsNull":false,"Value":0.1,"SubId":0}
,{"Dims":["10-20","F"],"IsNull":false,"Value":0.2,"SubId":0}
,{"Dims":["20-30","M"],"IsNull":false,"Value":0.3,"SubId":0}
,{"Dims":["20-30","F"],"IsNull":false,"Value":0.4,"SubId":0}
,{"Dims":["30-40","M"],"IsNull":false,"Value":0.5,"SubId":0}
,{"Dims":["30-40","F"],"IsNull":false,"Value":0.6,"SubId":0}
,{"Dims":["40+","M"],"IsNull":false,"Value":0.7,"SubId":0}
,{"Dims":["40+","F"],"IsNull":false,"Value":0.8,"SubId":0}
]
```

GET output table expression(s) from model run

Read a "page" of output table expression(s) values from model run.

Page is part of output table values defined by zero-based "start" row number and row count. If row count <= 0 then all rows returned.

Dimension(s) returned as enum codes or as string values if dimension type is simple (integer or boolean).

Methods:

```
GET /api/model/:model/run/:run/table/:name/expr
GET /api/model/:model/run/:run/table/:name/expr/start/:start
GET /api/model/:model/run/:run/table/:name/expr/start/:start/count/:count
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

```
:name - (required) output table name
```

```
:start - (optional) start "page" row number, zero-based.
:count - (optional) "page" size, number of rows to select.
```

By default oms service selects 100 rows (it can be configured). If count <= 0 specified then all rows selected.

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/expr
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/expr/start/2
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/expr/start/2/count/4
http://localhost:4040/api/model/_201208171604590148_/run/f172e98da17beb058f30f11768053456/table/salarySex/expr
http://localhost:4040/api/model/_201208171604590148_/run/2019_01_17_19_59_52_998/table/salarySex/expr
```

Return example:

```
[{"Dims":["L","M"],"Value":50,"IsNull":false,"ExprId":0}
,{"Dims":["L","F"],"Value":60,"IsNull":false,"ExprId":0}
,{"Dims":["L","all"],"Value":1,"IsNull":false,"ExprId":0}
,{"Dims":["M","M"],"Value":51.599999999999994,"IsNull":false,"ExprId":0}
,{"Dims":["M","F"],"Value":62,"IsNull":false,"ExprId":0}
,{"Dims":["M","all"],"Value":2,"IsNull":false,"ExprId":0}
,{"Dims":["H","M"],"Value":53.2,"IsNull":false,"ExprId":0}
,{"Dims":["H","F"],"Value":64,"IsNull":false,"ExprId":0}
,{"Dims":["H","all"],"Value":3,"IsNull":false,"ExprId":0}
,{"Dims":["L","M"],"Value":1,"IsNull":false,"ExprId":1}
,{"Dims":["L","F"],"Value":2,"IsNull":false,"ExprId":1}
,{"Dims":["L","all"],"Value":801,"IsNull":false,"ExprId":1}
,{"Dims":["M","M"],"Value":3,"IsNull":false,"ExprId":1}
,{"Dims":["M","F"],"Value":4,"IsNull":false,"ExprId":1}
,{"Dims":["M","all"],"Value":803,"IsNull":false,"ExprId":1}
,{"Dims":["H","M"],"Value":4,"IsNull":false,"ExprId":1}
,{"Dims":["H","F"],"Value":5,"IsNull":false,"ExprId":1}
,{"Dims":["H","all"],"Value":804,"IsNull":false,"ExprId":1}
,{"Dims":["L","M"],"Value":50,"IsNull":false,"ExprId":2}
,{"Dims":["L","F"],"Value":60,"IsNull":false,"ExprId":2}
,{"Dims":["L","all"],"Value":1,"IsNull":false,"ExprId":2}
,{"Dims":["M","M"],"Value":51.599999999999994,"IsNull":false,"ExprId":2}
,{"Dims":["M","F"],"Value":62,"IsNull":false,"ExprId":2}
,{"Dims":["M","all"],"Value":2,"IsNull":false,"ExprId":2}
,{"Dims":["H","M"],"Value":53.2,"IsNull":false,"ExprId":2}
,{"Dims":["H","F"],"Value":64,"IsNull":false,"ExprId":2}
,{"Dims":["H","all"],"Value":3,"IsNull":false,"ExprId":2}
,{"Dims":["L","M"],"Value":50,"IsNull":false,"ExprId":3}
,{"Dims":["L","F"],"Value":120,"IsNull":false,"ExprId":3}
,{"Dims":["L","all"],"Value":801,"IsNull":false,"ExprId":3}
,{"Dims":["M","M"],"Value":154.79999999999998,"IsNull":false,"ExprId":3}
,{"Dims":["M","F"],"Value":248,"IsNull":false,"ExprId":3}
,{"Dims":["M","all"],"Value":1606,"IsNull":false,"ExprId":3}
,{"Dims":["H","M"],"Value":212.8,"IsNull":false,"ExprId":3}
,{"Dims":["H","F"],"Value":320,"IsNull":false,"ExprId":3}
,{"Dims":["H","all"],"Value":2412,"IsNull":false,"ExprId":3}
]
```

GET output table accumulator(s) from model run

Read a "page" of output table accumulator(s) values from model run.

Page is part of output table values defined by zero-based "start" row number and row count. If row count <= 0 then all rows returned.

Dimension(s) returned as enum codes or as string values if dimension type is simple (integer or boolean).

Methods:

```
GET /api/model/:model/run/:run/table/:name/acc
GET /api/model/:model/run/:run/table/:name/acc/start/:start
GET /api/model/:model/run/:run/table/:name/acc/start/:start/count/:count
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

```
:name - (required) output table name
```

```
:start - (optional) start "page" row number, zero-based.
:count - (optional) "page" size, number of rows to select.
```

By default oms service selects 100 rows (it can be configured). If count <= 0 specified then all rows selected.

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/acc
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/acc/start/2
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/acc/start/2/count/4
http://localhost:4040/api/model/_201208171604590148_/run/f172e98da17beb058f30f11768053456/table/salarySex/acc
http://localhost:4040/api/model/_201208171604590148_/run/2019_01_17_19_59_52_998/table/salarySex/acc
```

Return example:


```
[{"Dims":["L","M"],"Value":50,"IsNull":false,"AccId":0,"SubId":0}
,{"Dims":["L","F"],"Value":60,"IsNull":false,"AccId":0,"SubId":0}
,{"Dims":["L","all"],"Value":1,"IsNull":false,"AccId":0,"SubId":0}
,{"Dims":["M","M"],"Value":51.599999999999994,"IsNull":false,"AccId":0,"SubId":0}
,{"Dims":["M","F"],"Value":62,"IsNull":false,"AccId":0,"SubId":0}
,{"Dims":["M","all"],"Value":2,"IsNull":false,"AccId":0,"SubId":0}
,{"Dims":["H","M"],"Value":53.2,"IsNull":false,"AccId":0,"SubId":0}
,{"Dims":["H","F"],"Value":64,"IsNull":false,"AccId":0,"SubId":0}
,{"Dims":["H","all"],"Value":3,"IsNull":false,"AccId":0,"SubId":0}
,{"Dims":["L","M"],"Value":1,"IsNull":false,"AccId":1,"SubId":0}
,{"Dims":["L","F"],"Value":2,"IsNull":false,"AccId":1,"SubId":0}
,{"Dims":["L","all"],"Value":801,"IsNull":false,"AccId":1,"SubId":0}
,{"Dims":["M","M"],"Value":3,"IsNull":false,"AccId":1,"SubId":0}
,{"Dims":["M","F"],"Value":4,"IsNull":false,"AccId":1,"SubId":0}
,{"Dims":["M","all"],"Value":803,"IsNull":false,"AccId":1,"SubId":0}
,{"Dims":["H","M"],"Value":4,"IsNull":false,"AccId":1,"SubId":0}
,{"Dims":["H","F"],"Value":5,"IsNull":false,"AccId":1,"SubId":0}
,{"Dims":["H","all"],"Value":804,"IsNull":false,"AccId":1,"SubId":0}
]
```

GET output table all accumulators from model run

Read a "page" of output table values from "all accumulators" view of model run.

"All accumulators" view include derived accumulators. Page is part of output table values defined by zero-based "start" row number and row count. If row count <= 0 then all rows returned.

Dimension(s) returned as enum codes or as string values if dimension type is simple (integer or boolean).

Methods:

```
GET /api/model/:model/run/:run/table/:name/all-acc
GET /api/model/:model/run/:run/table/:name/all-acc/start/:start
GET /api/model/:model/run/:run/table/:name/all-acc/start/:start/count/:count
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

```
:name - (required) output table name
```

```
:start - (optional) start "page" row number, zero-based.
:count - (optional) "page" size, number of rows to select.
```

By default oms service selects 100 rows (it can be configured). If count <= 0 specified then all rows selected.

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/all-acc
http://localhost:4040/api/model/modelOne_/run/Default/table/salarySex/all-acc/start/2
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/all-acc/start/2/count/4
http://localhost:4040/api/model/_201208171604590148_/run/f172e98da17beb058f30f11768053456/table/salarySex/all-acc
http://localhost:4040/api/model/_201208171604590148_/run/2019_01_17_19_59_52_998/table/salarySex/all-acc
```

Return example:


```
[{"Dims":["L","M"],"SubId":0,"IsNull":[false,false,false],"Value":[50,1,51]}
,{"Dims":["L","F"],"SubId":0,"IsNull":[false,false,false],"Value":[60,2,62]}
,{"Dims":["L","all"],"SubId":0,"IsNull":[false,false,false],"Value":[1,801,802]}
,{"Dims":["M","M"],"SubId":0,"IsNull":[false,false,false],"Value":[51.599999999999994,3,54.599999999999994]}
,{"Dims":["M","F"],"SubId":0,"IsNull":[false,false,false],"Value":[62,4,66]}
,{"Dims":["M","all"],"SubId":0,"IsNull":[false,false,false],"Value":[2,803,805]}
,{"Dims":["H","M"],"SubId":0,"IsNull":[false,false,false],"Value":[53.2,4,57.2]}
,{"Dims":["H","F"],"SubId":0,"IsNull":[false,false,false],"Value":[64,5,69]}
,{"Dims":["H","all"],"SubId":0,"IsNull":[false,false,false],"Value":[3,804,807]}
]
```

GET microdata values from model run

Read a "page" of microdata values from model run.

Page is part of microdata values defined by zero-based "start" row number and row count. If row count <= 0 then all rows returned.

Enum-based microdata attributes returned as enum codes.

Methods:

```
GET /api/model/:model/run/:run/microdata/:name/value
GET /api/model/:model/run/:run/microdata/:name/value/start/:start
GET /api/model/:model/run/:run/microdata/:name/value/start/:start/count/:count
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

```
:name - (required) microdata entity name
```

```
:start - (optional) start "page" row number, zero-based.
:count - (optional) "page" size, number of rows to select.
```

By default oms service selects 100 rows (it can be configured). If count <= 0 specified then all rows selected.

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/value
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/value/start/131040
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/value/start/131040/count/4
http://localhost:4040/api/model/_201208171604590148_/run/Microdata%20in%20database/microdata/Person/value
http://localhost:4040/api/model/_201208171604590148_/run/2019_01_17_19_59_52_998/microdata/Person/value
```

Return example:


```
[{
        "Key": 844424930164716,
        "Attr": [{
                "IsNull": false,
                "Value": "32"
            }, {
                "IsNull": false,
                "Value": "30-40"
            }, {
                "IsNull": false,
                "Value": "M"
            }, {
                "IsNull": false,
                "Value": "268271632"
            }, {
                "IsNull": false,
                "Value": "201203724"
            }, {
                "IsNull": false,
                "Value": "H"
            }, {
                "IsNull": false,
                "Value": "Full"
            }, {
                "IsNull": false,
```

```
            "IsNull": false,
            "Value": "false"
        }, {
            "IsNull": false,
            "Value": "0"
        }
    ]
}, {
    "Key": 844424930164717,
    "Attr": [{
            "IsNull": false,
            "Value": "23"
        }, {
            "IsNull": false,
            "Value": "20-30"
        }, {
            "IsNull": false,
            "Value": "F"
        }, {
            "IsNull": false,
            "Value": "268279823"
        }, {
            "IsNull": false,
            "Value": "201209867.25"
        }, {
            "IsNull": false,
            "Value": "H"
        }, {
            "IsNull": false,
            "Value": "Full"
        }, {
            "IsNull": false,
            "Value": "false"
        }, {
            "IsNull": false,
            "Value": "0"
        }
    ]
}, {
    "Key": 844424930164718,
    "Attr": [{
            "IsNull": false,
            "Value": "14"
        }, {
            "IsNull": false,
            "Value": "10-20"
        }, {
            "IsNull": false,
            "Value": "M"
        }, {
            "IsNull": false,
            "Value": "0"
        }, {
            "IsNull": false,
            "Value": "0"
        }, {
            "IsNull": false,
            "Value": "L"
        }, {
            "IsNull": false,
            "Value": "Part"
        }, {
            "IsNull": false,
            "Value": "false"
        }, {
            "IsNull": false,
            "Value": "0"
        }
    ]
}, {
    "Key": 844424930164719,
    "Attr": [{
            "IsNull": false,
            "Value": "5"
        }, {
            "IsNull": false,
            "Value": "10-20"
        }, {
            "IsNull": false,
            "Value": "F"
        }, {
            "IsNull": false,
            "Value": "0"
        }, {
            "IsNull": false,
            "Value": "0"
        }, {
            "IsNull": false,
```

```
            "Value": "L"
        }, {
            "IsNull": false,
            "Value": "Part"
        }, {
            "IsNull": false,
            "Value": "false"
        }, {
            "IsNull": false,
            "Value": "0"
        }
    ]
  }
]
```

GET csv parameter values from workset

Read entire parameter values from workset as csv file.

Response stream is UTF-8 parameter.csv file attachment, optionally starts with byte order mark (BOM).

Dimension(s) and enum-based parameters returned as enum codes.

Methods:

```
GET /api/model/:model/workset/:set/parameter/:name/csv
GET /api/model/:model/workset/:set/parameter/:name/csv-bom
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:set - (required) workset name
```

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default".

```
:name - (required) parameter name
```

Call examples:

```
http://localhost:4040/api/model/modelOne/workset/modelOne_other/parameter/ageSex/csv
http://localhost:4040/api/model/modelOne/workset/modelOne_other/parameter/ageSex/csv-bom
```

Return example:

```
sub_id,dim0,dim1,param_value
0,10-20,M,1.1
0,10-20,F,1.2
0,20-30,M,1.3
0,20-30,F,1.4
0,30-40,M,1.5
0,30-40,F,1.6
0,40+,M,1.7
0,40+,F,1.8
```

GET csv parameter values from workset (enum id's)

Read entire parameter values from workset as csv file.

Response stream is UTF-8 parameter.csv file attachment, optionally starts with byte order mark (BOM).

Dimension(s) and enum-based parameters returned as enum id, not enum codes.

Methods:

```
GET /api/model/:model/workset/:set/parameter/:name/csv-id
GET /api/model/:model/workset/:set/parameter/:name/csv-id-bom
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:set - (required) workset name
```

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default".

```
:name - (required) parameter name
```

Call examples:

```
http://localhost:4040/api/model/modelOne/workset/modelOne_other/parameter/ageSex/csv-id
http://localhost:4040/api/model/modelOne/workset/modelOne_other/parameter/ageSex/csv-id-bom
```

Return example:

```
sub_id,dim0,dim1,param_value
0,10,0,1.1
0,10,1,1.2
0,20,0,1.3
0,20,1,1.4
0,30,0,1.5
0,30,1,1.6
0,40,0,1.7
0,40,1,1.8
```

GET csv parameter values from model run

Read entire parameter values from model run as csv file.

Response stream is UTF-8 parameter.csv file attachment, optionally starts with byte order mark (BOM).

Dimension(s) and enum-based parameters returned as enum codes.

Methods:

```
GET /api/model/:model/run/:run/parameter/:name/csv
GET /api/model/:model/run/:run/parameter/:name/csv-bom
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

```
:name - (required) parameter name
```

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Default/parameter/ageSex/csv
http://localhost:4040/api/model/modelOne/run/Default/parameter/ageSex/csv-bom
http://localhost:4040/api/model/modelOne/run/f172e98da17beb058f30f11768053456/parameter/ageSex/csv
http://localhost:4040/api/model/modelOne/run/f172e98da17beb058f30f11768053456/parameter/ageSex/csv-bom
http://localhost:4040/api/model/modelOne/run/2019_01_17_19_59_52_998/parameter/ageSex/csv
```

Return example:

```
sub_id,dim0,dim1,param_value
0,10-20,M,0.1
0,10-20,F,0.2
0,20-30,M,0.3
0,20-30,F,0.4
0,30-40,M,0.5
0,30-40,F,0.6
0,40+,M,0.7
0,40+,F,0.8
```

GET csv parameter values from model run (enum id's)

Read entire parameter values from model run as csv file.

Response stream is UTF-8 parameter.csv file attachment, optionally starts with byte order mark (BOM).

Dimension(s) and enum-based parameters returned as enum id, not enum codes.

Methods:

```
GET /api/model/:model/run/:run/parameter/:name/csv-id
GET /api/model/:model/run/:run/parameter/:name/csv-id-bom
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

```
:name - (required) parameter name
```

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Default/parameter/ageSex/csv-id
http://localhost:4040/api/model/modelOne/run/Default/parameter/ageSex/csv-id-bom
http://localhost:4040/api/model/modelOne/run/f172e98da17beb058f30f11768053456/parameter/ageSex/csv-id
http://localhost:4040/api/model/modelOne/run/f172e98da17beb058f30f11768053456/parameter/ageSex/csv-id-bom
http://localhost:4040/api/model/modelOne/run/2019_01_17_19_59_52_998/parameter/ageSex/csv-id
```

Return example:

```
sub_id,dim0,dim1,param_value
0,10,0,0.1
0,10,1,0.2
0,20,0,0.3
0,20,1,0.4
0,30,0,0.5
0,30,1,0.6
0,40,0,0.7
0,40,1,0.8
```

GET csv output table expressions from model run

Read entire output table expression(s) values from model run as csv file.

Response stream is UTF-8 outputTable.csv file attachment, optionally starts with byte order mark (BOM).

Dimension(s) returned as enum codes.

Methods:

```
GET /api/model/:model/run/:run/table/:name/expr/csv
GET /api/model/:model/run/:run/table/:name/expr/csv-bom
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

```
:name - (required) output table name
```

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/expr/csv
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/expr/csv-bom
http://localhost:4040/api/model/_201208171604590148_/run/f172e98da17beb058f30f11768053456/table/salarySex/expr/csv
http://localhost:4040/api/model/_201208171604590148_/run/2019_01_17_19_59_52_998/table/salarySex/expr/csv
```

Return example:

```
expr_name,dim0,dim1,expr_value
expr0,L,M,50
expr0,L,F,60
expr0,L,all,1
expr0,M,M,51.6
expr0,M,F,62
expr0,M,all,2
expr0,H,M,53.2
expr0,H,F,64
expr0,H,all,3
expr1,L,M,1
expr1,L,F,2
expr1,L,all,801
expr1,M,M,3
expr1,M,F,4
expr1,M,all,803
expr1,H,M,4
expr1,H,F,5
expr1,H,all,804
expr2,L,M,50
expr2,L,F,60
expr2,L,all,1
expr2,M,M,51.6
expr2,M,F,62
expr2,M,all,2
expr2,H,M,53.2
expr2,H,F,64
expr2,H,all,3
expr3,L,M,50
expr3,L,F,120
expr3,L,all,801
expr3,M,M,154.8
expr3,M,F,248
expr3,M,all,1606
expr3,H,M,212.8
expr3,H,F,320
expr3,H,all,2412
```

GET csv output table expressions from model run (enum id's)

Read entire output table expression(s) values from model run as csv file.

Response stream is UTF-8 outputTable.csv file attachment, optionally starts with byte order mark (BOM).

Dimension(s) returned as enum id's.

Methods:

```
GET /api/model/:model/run/:run/table/:name/expr/csv-id
GET /api/model/:model/run/:run/table/:name/expr/csv-id-bom
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

```
:name - (required) output table name
```

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/expr/csv-id
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/expr/csv-id-bom
http://localhost:4040/api/model/_201208171604590148_/run/f172e98da17beb058f30f11768053456/table/salarySex/expr/csv-id
http://localhost:4040/api/model/_201208171604590148_/run/2019_01_17_19_59_52_998/table/salarySex/expr/csv-id
```

Return example:

```
expr_id,dim0,dim1,expr_value
0,100,0,50
0,100,1,60
0,100,800,1
0,200,0,51.6
0,200,1,62
0,200,800,2
0,300,0,53.2
0,300,1,64
0,300,800,3
1,100,0,1
1,100,1,2
1,100,800,801
1,200,0,3
1,200,1,4
1,200,800,803
1,300,0,4
1,300,1,5
1,300,800,804
2,100,0,50
2,100,1,60
2,100,800,1
2,200,0,51.6
2,200,1,62
2,200,800,2
2,300,0,53.2
2,300,1,64
2,300,800,3
3,100,0,50
3,100,1,120
3,100,800,801
3,200,0,154.8
3,200,1,248
3,200,800,1606
3,300,0,212.8
3,300,1,320
3,300,800,2412
```

GET csv output table accumulators from model run

Read entire output table accumulator(s) values from model run as csv file.

Response stream is UTF-8 outputTable.csv file attachment, optionally starts with byte order mark (BOM).

Dimension(s) returned as enum codes.

Methods:

```
GET /api/model/:model/run/:run/table/:name/acc/csv
GET /api/model/:model/run/:run/table/:name/acc/csv-bom
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

```
:name - (required) output table name
```

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/acc/csv
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/acc/csv-bom
http://localhost:4040/api/model/_201208171604590148_/run/f172e98da17beb058f30f11768053456/table/salarySex/acc/csv
http://localhost:4040/api/model/_201208171604590148_/run/2019_01_17_19_59_52_998/table/salarySex/acc/csv
```

Return example:

```
acc_name,sub_id,dim0,dim1,acc_value
acc0,0,L,M,50
acc0,0,L,F,60
acc0,0,L,all,1
acc0,0,M,M,51.6
acc0,0,M,F,62
acc0,0,M,all,2
acc0,0,H,M,53.2
acc0,0,H,F,64
acc0,0,H,all,3
acc1,0,L,M,1
acc1,0,L,F,2
acc1,0,L,all,801
acc1,0,M,M,3
acc1,0,M,F,4
acc1,0,M,all,803
acc1,0,H,M,4
acc1,0,H,F,5
acc1,0,H,all,804
```

GET csv output table accumulators from model run (enum id's)

Read entire output table accumulator(s) values from model run as csv file.

Response stream is UTF-8 outputTable.csv file attachment, optionally starts with byte order mark (BOM).

Dimension(s) returned as enum id's.

Methods:

```
GET /api/model/:model/run/:run/table/:name/acc/csv-id
GET /api/model/:model/run/:run/table/:name/acc/csv-id-bom
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

```
:name - (required) output table name
```

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/acc/csv-id
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/acc/csv-id-bom
http://localhost:4040/api/model/_201208171604590148_/run/f172e98da17beb058f30f11768053456/table/table/salarySex/acc/csv-id
http://localhost:4040/api/model/_201208171604590148_/run/2019_01_17_19_59_52_998/table/table/salarySex/acc/csv-id
```

Return example:

```
acc_id,sub_id,dim0,dim1,acc_value
0,0,100,0,50
0,0,100,1,60
0,0,100,800,1
0,0,200,0,51.6
0,0,200,1,62
0,0,200,800,2
0,0,300,0,53.2
0,0,300,1,64
0,0,300,800,3
1,0,100,0,1
1,0,100,1,2
1,0,100,800,801
1,0,200,0,3
1,0,200,1,4
1,0,200,800,803
1,0,300,0,4
1,0,300,1,5
1,0,300,800,804
```

GET csv output table all accumulators from model run

Read entire output table "all-accumulators" view values from model run as csv file.

Response stream is UTF-8 outputTable.csv file attachment, optionally starts with byte order mark (BOM).

Dimension(s) returned as enum codes.

Methods:

```
GET /api/model/:model/run/:run/table/:name/all-acc/csv
GET /api/model/:model/run/:run/table/:name/all-acc/csv-bom
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

```
:name - (required) output table name
```

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/all-acc/csv
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/all-acc/csv-bom
http://localhost:4040/api/model/_201208171604590148_/run/f172e98da17beb058f30f11768053456/table/salarySex/all-acc/csv
http://localhost:4040/api/model/modelOne/run/2019_01_17_19_59_52_998/table/salarySex/all-acc/csv
```

Return example:

```
sub_id,dim0,dim1,acc0,acc1,acc2
0,L,M,50,1,51
0,L,F,60,2,62
0,L,all,1,801,802
0,M,M,51.6,3,54.6
0,M,F,62,4,66
0,M,all,2,803,805
0,H,M,53.2,4,57.2
0,H,F,64,5,69
0,H,all,3,804,807
```

GET csv output table all accumulators from model run (enum id's)

Read entire output table "all-accumulators" view values from model run as csv file.

Response stream is UTF-8 outputTable.csv file attachment, optionally starts with byte order mark (BOM).

Dimension(s) returned as enum id's.

Methods:

```
GET /api/model/:model/run/:run/table/:name/all-acc/csv-id
GET /api/model/:model/run/:run/table/:name/all-acc/csv-id-bom
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

```
:name - (required) output table name
```

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/all-acc/csv-id
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/all-acc/csv-id-bom
http://localhost:4040/api/model/_201208171604590148_/run/f172e98da17beb058f30f11768053456/table/salarySex/all-acc/csv-id
http://localhost:4040/api/model/modelOne/run/2019_01_17_19_59_52_998/table/salarySex/all-acc/csv-id
```

Return example:

```
sub_id,dim0,dim1,acc0,acc1,acc2
0,100,0,50,1,51
0,100,1,60,2,62
0,100,800,1,801,802
0,200,0,51.6,3,54.6
0,200,1,62,4,66
0,200,800,2,803,805
0,300,0,53.2,4,57.2
0,300,1,64,5,69
0,300,800,3,804,807
```

GET csv microdata values from model run

Read entire microdata values from model run as csv file.

Response stream is UTF-8 microdata.csv file attachment, optionally starts with byte order mark (BOM).

Enum-based microdata attributes returned as enum codes.

Methods:

```
GET /api/model/:model/run/:run/microdata/:name/csv
GET /api/model/:model/run/:run/microdata/:name/csv-bom
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

```
:name - (required) microdata entity name
```

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/csv
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/csv-bom
http://localhost:4040/api/model/modelOne/run/f172e98da17beb058f30f11768053456/microdata/Person/csv-bom
http://localhost:4040/api/model/modelOne/run/2019_01_17_19_59_52_998/microdata/Person/csv
http://localhost:4040/api/model/_201208171604590148_/run/Microdata%20in%20database/microdata/Person/csv
```

Return example:

```
key,Age,AgeGroup,Sex,Income,Salary,SalaryGroup,FullTime,IsOldAge,Pension
0,91,40+,F,23000,0,L,Part,true,23000
1,82,40+,M,23000,0,L,Part,true,23000
2,73,40+,F,29900,6900,L,Part,true,23000
3,64,40+,M,32764,24573,L,Full,false,0
................
................
844424930164728,24,20-30,M,268369924,201277443,H,Full,false,0
844424930164729,15,10-20,F,0,0,L,Part,false,0
844424930164730,6,10-20,M,0,0,L,Part,false,0
844424930164731,97,40+,F,53678899.4,0,L,Part,true,53678899.4
```

GET csv microdata values from model run (enum id's)

Read entire microdata values from model run as csv file.

Response stream is UTF-8 microdata.csv file attachment, optionally starts with byte order mark (BOM).

Enum-based microdata attributes returned as enum id, not enum codes.

Methods:

```
GET /api/model/:model/run/:run/microdata/:name/csv-id
GET /api/model/:model/run/:run/microdata/:name/csv-id-bom
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

```
:name - (required) microdata entity name
```

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/csv-id
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/csv-id-bom
http://localhost:4040/api/model/modelOne/run/f172e98da17beb058f30f11768053456/microdata/Person/csv-id-bom
http://localhost:4040/api/model/modelOne/run/2019_01_17_19_59_52_998/microdata/Person/csv-id
http://localhost:4040/api/model/_201208171604590148_/run/Microdata%20in%20database/microdata/Person/csv-id
```

Return example:

```
key,Age,AgeGroup,Sex,Income,Salary,SalaryGroup,FullTime,IsOldAge,Pension
0,91,40,1,23000,0,100,33,true,23000
1,82,40,0,23000,0,100,33,true,23000
2,73,40,1,29900,6900,100,33,true,23000
3,64,40,0,32764,24573,100,22,false,0
................
................
844424930164728,24,20,0,268369924,201277443,300,22,false,0
844424930164729,15,10,1,0,0,100,33,false,0
844424930164730,6,10,0,0,0,100,33,false,0
844424930164731,97,40,1,53678899.4,0,100,33,true,53678899.4
```

GET list of modeling tasks

Get list of modeling tasks: language-neutral part of task list metadata.

Methods:

```
GET /api/model/:model/task-list
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

Call examples from browser:

```
http://localhost:4040/api/model/modelOne/task-list
http://localhost:4040/api/model/_201208171604590148_/task-list
```

Return example:

```
[
  {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "taskOne",
    "Txt": [],
    "Set": [],
    "TaskRun": []
  }
]
```

GET list of modeling tasks including text (description and notes)

Get list of modeling tasks, including text (description and notes).

Methods:

```
GET /api/model/:model/task-list/text
GET /api/model/:model/task-list/text/lang/:lang
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:lang - (optional) language code
```

If optional `lang` argument specified then result in that language else in browser language. If no such language exist then text portion of result (description and notes) is empty.

Call examples from browser:

```
http://localhost:4040/api/model/modelOne/task-list/text
http://localhost:4040/api/model/modelOne/task-list/text/lang/EN
```

Return example:


```
[
  {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "taskOne",
    "Txt": [
      {
        "LangCode": "EN",
        "Descr": "Task One for Model One",
        "Note": "Task One: two set of input parameters"
      }
    ],
    "Set": [],
    "TaskRun": []
  }
]
```

GET modeling task input worksets

Get modeling task input sets: language-neutral part of task metadata.

Methods:

```
GET /api/model/:model/task/:task/sets
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:task - (required) modeling task name
```

Task is uniquely identified by name (inside the model). Different models can have tasks with same name, i.e. each model can have task with name "My First Task".

Call examples from browser:

```
http://localhost:4040/api/model/modelOne/task/taskOne/sets
http://localhost:4040/api/model/_201208171604590148_/task/taskOne/sets
```

Return example:


```
{
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "taskOne",
  "Txt": [],
  "Set": [
    "Default",
    "modelOne_other"
  ],
  "TaskRun": []
}
```

GET modeling task run history

Get modeling task run history: language-neutral part of task metadata.

Methods:

```
GET /api/model/:model/task/:task/runs
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:task - (required) modeling task name
```

Task is uniquely identified by name (inside the model). Different models can have tasks with same name, i.e. each model can have task with name "My First Task".

Call examples from browser:

```
http://localhost:4040/api/model/modelOne/task/taskOne/runs
http://localhost:4040/api/model/modelOne/task/taskOne/runs
```

Return example:


```
{
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "taskOne",
  "Txt": [],
  "Set": [],
  "TaskRun": [
    {
      "Name": "First Task Run",
      "SubCount": 1,
      "CreateDateTime": "2020-03-24 16:29:20.427",
      "Status": "s",
      "UpdateDateTime": "2020-03-24 16:29:20.857",
      "RunStamp": "2020_03_24_16_29_20_427",
      "TaskRunSet": [
        {
          "Run": {
            "Name": "First_Task_Run_Default",
            "SubCompleted": 1,
            "CreateDateTime": "2020-03-24 16:29:20.459",
            "Status": "s",
            "RunDigest": "aa3bed04d833966853bdf04f841c5feb",
            "ValueDigest": "6c5c0f48e19f67899c868688bb8a23fd",
            "RunStamp": "2020_03_24_16_29_20_427"
          },
          "SetName": "Default"
        },
        {
          "Run": {
            "Name": "First_Task_Run_modelOne_other",
            "SubCompleted": 1,
            "CreateDateTime": "2020-03-24 16:29:20.667",
            "Status": "s",
            "RunDigest": "f8e078c414f15c79d19a2666b126dea5",
            "ValueDigest": "fb27d108fae2040fa1cae6f49704a1b7",
            "RunStamp": "2020_03_24_16_29_20_427"
          },
          "SetName": "modelOne_other"
        }
      ]
    }
  ]
}
```

GET status of modeling task run

Get status of modeling task run.

Methods:

```
GET /api/model/:model/task/:task/run-status/run/:run
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:task - (required) modeling task name
```

Task is uniquely identified by name (inside the model). Different models can have tasks with same name, i.e. each model can have task with name "My First Task".

```
:run - (required) modeling task run stamp or task run name
```

Task run stamp and task run name can be explicitly specified as model run options. If task run stamp not specified then it automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is recommended to specify unique run stamp or run name when you are running the modeling task. If task run stamp or task run name is not unique then result of this call is undefined. You can use GET status of modeling task run list method to get status of multiple runs with same name or stamp.

Call examples from browser:

```
http://localhost:4040/api/model/modelOne/task/taskOne/run-status/run/First%20Task%20Run
http://localhost:4040/api/model/_201208171604590148_/task/taskOne/run-status/run/2019_01_17_19_59_53_260
```

Return example: *This is a beta version and may change in the future.*


```
{
  "TaskRunId": 101,
  "TaskId": 1,
  "Name": "First Task Run",
  "SubCount": 1,
  "CreateDateTime": "2019-01-17 19:59:53.260",
  "Status": "s",
  "UpdateDateTime": "2019-01-17 19:59:53.539",
  "RunStamp": "2019_01_17_19_59_53_260"
}
```

GET status of modeling task run list

Get status of modeling task runs.

Methods:

```
GET /api/model/:model/task/:task/run-status/list/:run
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:task - (required) modeling task name
```

Task is uniquely identified by name (inside the model). Different models can have tasks with same name, i.e. each model can have task with name "My First Task".

```
:run - (required) modeling task run stamp or task run name
```

Task run stamp and task run name can be explicitly specified as model run options. If task run stamp not specified then it automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is recommended to specify unique run stamp or run name when you are running the modeling task.

Call examples from browser:

```
http://localhost:4040/api/model/modelOne/task/taskOne/run-status/list/First%20Task%20Run
http://localhost:4040/api/model/_201208171604590148_/task/taskOne/run-status/list/2019_01_17_19_59_53_260
```

Return example: *This is a beta version and may change in the future.*


```
[
  {
    "TaskRunId": 101,
    "TaskId": 1,
    "Name": "First Task Run",
    "SubCount": 1,
    "CreateDateTime": "2020-02-01 12:10:45.090",
    "Status": "s",
    "UpdateDateTime": "2020-02-01 12:10:45.523",
    "RunStamp": "2020_02_01_12_10_45_090"
  }
]
```

GET status of modeling task first run

Get status of modeling task first run.

Methods:

```
GET /api/model/:model/task/:task/run-status/first
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:task - (required) modeling task name
```

Task is uniquely identified by name (inside the model). Different models can have tasks with same name, i.e. each model can have task with name "My First Task".

Call examples from browser:

```
http://localhost:4040/api/model/modelOne/task/taskOne/run-status/first
http://localhost:4040/api/model/_201208171604590148_/task/taskOne/run-status/first
```

Return example: *This is a beta version and may change in the future.*


```
{
  "TaskRunId": 101,
  "TaskId": 1,
  "Name": "First Task Run",
  "SubCount": 1,
  "CreateDateTime": "2019-01-17 19:59:53.260",
  "Status": "s",
  "UpdateDateTime": "2019-01-17 19:59:53.539",
  "RunStamp": "2019_01_17_19_59_53_260"
}
```

GET status of modeling task last run

Get status of modeling task last (most recent) run.

Methods:

```
GET /api/model/:model/task/:task/run-status/last
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:task - (required) modeling task name
```

Task is uniquely identified by name (inside the model). Different models can have tasks with same name, i.e. each model can have task with name "My First Task".

Call examples from browser:

```
http://localhost:4040/api/model/modelOne/task/taskOne/run-status/last
http://localhost:4040/api/model/_201208171604590148_/task/taskOne/run-status/last
```

Return example: *This is a beta version and may change in the future.*

```
{
  "TaskRunId": 101,
  "TaskId": 1,
  "Name": "First Task Run",
  "SubCount": 1,
  "CreateDateTime": "2019-01-17 19:59:53.260",
  "Status": "s",
  "UpdateDateTime": "2019-01-17 19:59:53.539",
  "RunStamp": "2019_01_17_19_59_53_260"
}
```

GET status of modeling task last completed run

Get status of modeling task last completed run. Run completed if run status one of: s=success, x=exit, e=error.

Methods:

```
GET /api/model/:model/task/:task/run-status/last-completed
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:task - (required) modeling task name
```

Task is uniquely identified by name (inside the model). Different models can have tasks with same name, i.e. each model can have task with name "My First Task".

Call examples from browser:

```
http://localhost:4040/api/model/modelOne/task/taskOne/run-status/last-completed
http://localhost:4040/api/model/_201208171604590148_/task/taskOne/run-status/last-completed
```

Return example: *This is a beta version and may change in the future.*


```
{
  "TaskRunId": 101,
  "TaskId": 1,
  "Name": "First Task Run",
  "SubCount": 1,
  "CreateDateTime": "2019-01-17 19:59:53.260",
  "Status": "s",
  "UpdateDateTime": "2019-01-17 19:59:53.539",
  "RunStamp": "2019_01_17_19_59_53_260"
}
```

GET modeling task including text (description and notes)

Get modeling task and task run history, including text (description and notes)

Methods:

```
GET /api/model/:model/task/:task/text
GET /api/model/:model/task/:task/text/lang/:lang
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:task - (required) modeling task name
```

Task is uniquely identified by name (inside the model). Different models can have tasks with same name, i.e. each model can have task with name "My First Task".

```
:lang - (optional) language code
```

If optional `lang` argument specified then result in that language else in browser language. If no such language exist then text portion of result (description and notes) is empty.

Call examples from browser:

```
http://localhost:4040/api/model/modelOne/task/taskOne/text
http://localhost:4040/api/model/modelOne/task/taskOne/text/lang/EN
http://localhost:4040/api/model/_201208171604590148_/task/taskOne/text/lang/EN
```

Return example:


```
{
  "Task": {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "taskOne",
    "Txt": [
      {
        "LangCode": "EN",
        "Descr": "Task One for Model One",
        "Note": "Task One: two set of input parameters"
      }
    ],
    "Set": [
      "Default",
      "modelOne_other"
    ],
    "TaskRun": [
      {
        "Name": "First Task Run",
        "SubCount": 1,
        "CreateDateTime": "2020-03-24 16:29:20.427",
        "Status": "s",
        "UpdateDateTime": "2020-03-24 16:29:20.857",
        "RunStamp": "2020_03_24_16_29_20_427",
        "TaskRunSet": [
          {
            "Run": {
              "Name": "First_Task_Run_Default",
              "SubCompleted": 1,
              "CreateDateTime": "2020-03-24 16:29:20.459",
              "Status": "s",
              "RunDigest": "aa3bed04d833966853bdf04f841c5feb",
              "ValueDigest": "6c5c0f48e19f67899c868688bb8a23fd",
              "RunStamp": "2020_03_24_16_29_20_427"
            },
            "SetName": "Default"
          },
          {
            "Run": {
```

```
            "Run": {
              "Name": "First_Task_Run_modelOne_other",
              "SubCompleted": 1,
              "CreateDateTime": "2020-03-24 16:29:20.667",
              "Status": "s",
              "RunDigest": "f8e078c414f15c79d19a2666b126dea5",
              "ValueDigest": "fb27d108fae2040fa1cae6f49704a1b7",
              "RunStamp": "2020_03_24_16_29_20_427"
            },
            "SetName": "modelOne_other"
          }
        ]
      }
    ]
  },
  "Txt": {
    "SetTxt": {
      "Default": [
        {
          "LangCode": "EN",
          "Descr": "Model One default set of parameters",
          "Note": ""
        }
      ],
      "modelOne_other": [
        {
          "LangCode": "EN",
          "Descr": "Model One other set of parameters",
          "Note": ""
        }
      ]
    },
    "RunTxt": {
      "aa3bed04d833966853bdf04f841c5feb": [
        {
          "LangCode": "EN",
          "Descr": "Model One default set of parameters",
          "Note": ""
        }
      ],
      "f8e078c414f15c79d19a2666b126dea5": [
        {
          "LangCode": "EN",
          "Descr": "Model One other set of parameters",
          "Note": ""
        }
      ]
    }
  }
}
```

GET modeling task text in all languages

Get modeling task and task run history, including text (description and notes) in all languages.

Methods:

```
GET /api/model/:model/task/:task/text/all
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:task - (required) modeling task name
```

Task is uniquely identified by name (inside the model). Different models can have tasks with same name, i.e. each model can have task with name "My First Task".

Call examples from browser:

```
http://localhost:4040/api/model/modelOne/task/taskOne/text/all
http://localhost:4040/api/model/_201208171604590148_/task/taskOne/text/all
```

Return example:


```
{
  "Task": {
    "ModelName": "modelOne",
    "ModelDigest": "_201208171604590148_",
    "Name": "taskOne",
    "Txt": [
      {
        "LangCode": "EN",
        "Descr": "Task One for Model One",
        "Note": "Task One: two set of input parameters"
      },
      {
        "LangCode": "FR",
        "Descr": "(FR) Task One for Model One",
        "Note": ""
      }
    ],
    "Set": [
      "Default",
      "modelOne_other"
    ],
    "TaskRun": [
      {
        "Name": "First Task Run",
        "SubCount": 1,
        "CreateDateTime": "2020-03-24 16:29:20.427",
        "Status": "s",
        "UpdateDateTime": "2020-03-24 16:29:20.857",
        "RunStamp": "2020_03_24_16_29_20_427",
        "TaskRunSet": [
          {
            "Run": {
              "Name": "First_Task_Run_Default",
              "SubCompleted": 1,
              "CreateDateTime": "2020-03-24 16:29:20.459",
              "Status": "s",
              "RunDigest": "aa3bed04d833966853bdf04f841c5feb",
              "ValueDigest": "6c5c0f48e19f67899c868688bb8a23fd",
              "RunStamp": "2020_03_24_16_29_20_427"
            },
            "SetName": "Default"
          },
          {
            "Run": {
              "Name": "First_Task_Run_modelOne_other",
              "SubCompleted": 1,
              "CreateDateTime": "2020-03-24 16:29:20.667",
              "Status": "s",
```

```
        "RunDigest": "f8e078c414f15c79d19a2666b126dea5",
        "ValueDigest": "fb27d108fae2040fa1cae6f49704a1b7",
        "RunStamp": "2020_03_24_16_29_20_427"
       },
       "SetName": "modelOne_other"
      }
     ]
    }
   ]
  },
  "Txt": {
   "SetTxt": {
    "Default": [
     {
      "LangCode": "EN",
      "Descr": "Model One default set of parameters",
      "Note": ""
     },
     {
      "LangCode": "FR",
      "Descr": "(FR) Model One default set of parameters",
      "Note": ""
     }
    ],
    "modelOne_other": [
     {
      "LangCode": "EN",
      "Descr": "Model One other set of parameters",
      "Note": ""
     },
     {
      "LangCode": "FR",
      "Descr": "(FR) Model One other set of parameters",
      "Note": ""
     }
    ]
   },
   "RunTxt": {
    "aa3bed04d833966853bdf04f841c5feb": [
     {
      "LangCode": "EN",
      "Descr": "Model One default set of parameters",
      "Note": ""
     },
     {
      "LangCode": "FR",
      "Descr": "(FR) Model One default set of parameters",
      "Note": ""
     }
    ],
    "f8e078c414f15c79d19a2666b126dea5": [
     {
      "LangCode": "EN",
      "Descr": "Model One other set of parameters",
      "Note": ""
     },
     {
      "LangCode": "FR",
      "Descr": "(FR) Model One other set of parameters",
      "Note": ""
     }
    ]
   }
  }
 }
```

PATCH create or replace profile

Create new or replace existing profile.

Profile is a set of key-value options which can be used to run the model.

This method insert new or replace all existing profile options. If no such profile exist in database then new profile created.

If profile already exist then it is delete-insert operation:

- all existing profile key-value options deleted from database;
- new key-value options inserted.

Profile is uniquely identified by name (inside of the database). Profile are not a property of the model and you can use same profile for different models. *Beta version: beta version API uses model name or digest to find profile.*

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

This is a beta version and may change in the future.

Method:

```
PATCH /api/model/:model/profile
```

For example:

```
curl -v -X PATCH -H "Content-Type: application/json" "http://localhost:4040/api/model/modelOne/profile" -d @m1.profile.json
```

JSON arguments:

It is expected to be same JSON as return of GET model profile method.

For example (m1.profile.json file):

```
{
  "Name": "m1",
  "Opts": {
    "OpenM.SparseOutput": "false",
    "Parameter.StartingSeed": "192"
  }
}
```

DELETE profile

Delete existing profile.

Profile is a set of key-value options which can be used to run the model.

This method does delete of existing profile and all profile options. If no such profile exist in database then nothing is changed (it is no-op).

Profile is uniquely identified by name (inside of the database). Profile are not a property of the model and you can use same profile for different models. *Beta version: beta version API uses model name or digest to find profile.*

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

This is a beta version and may change in the future.

Method:

```
DELETE /api/model/:model/profile/:profile
```

For example:

```
curl -v -X DELETE http://localhost:4040/api/model/modelOne/profile/m1
curl -v -X DELETE http://localhost:4040/api/model/_201208171604590148_/profile/m1
```

```
curl -v -X DELETE http://localhost:4040/api/model/modelOne/profile/m1

*   Trying ::1...
* TCP_NODELAY set
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 4040 (#0)
> DELETE /api/model/modelOne/profile/m1 HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.54.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Location: /api/model/modelOne/profile/m1
< Date: Fri, 28 Dec 2018 03:06:21 GMT
< Content-Length: 0
<
* Connection #0 to host localhost left intact
```

POST create or replace profile option

Create new or replace existing profile option.

Profile is a set of key-value options which can be used to run the model.

This method insert new or replace all existing profile key-value options. If no such profile exist in database then new profile created. If no such key exist in that profile then new key-value inserted. If key already exist the value replaced.

Profile is uniquely identified by name (inside of the database). Profile are not a property of the model and you can use same profile for different models. *Beta version: beta version API uses model name or digest to find profile.*

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

This is a beta version and may change in the future.

Method:

```
POST /api/model/:model/profile/:profile/key/:key/value/:value
```

For example:

```
curl -v -X POST http://localhost:4040/api/model/modelOne/profile/m1/key/Parameter.StartingSeed/value/4095

*   Trying ::1...
* TCP_NODELAY set
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 4040 (#0)
> PATCH /api/model/modelOne/profile/m1/key/Parameter.StartingSeed/value/4095 HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.54.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Location: /api/model/modelOne/profile/m1/key/Parameter.StartingSeed
< Date: Fri, 28 Dec 2018 03:10:49 GMT
< Content-Length: 0
<
* Connection #0 to host localhost left intact
```

DELETE profile option

Delete existing profile option.

Profile is a set of key-value options which can be used to run the model.

This method does delete of existing profile key and associated value. If no such option key exist in that profile then nothing is changed (it is no-op).

Profile is uniquely identified by name (inside of the database). Profile are not a property of the model and you can use same profile for different models. *Beta version: beta version API uses model name or digest to find profile.*

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

This is a beta version and may change in the future.

Method:

```
DELETE /api/model/:model/profile/:profile/key/:key
```

For example:

```
curl -v -X DELETE http://localhost:4040/api/model/modelOne/profile/m1/key/Parameter.StartingSeed

*   Trying ::1...
* TCP_NODELAY set
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 4040 (#0)
> DELETE /api/model/modelOne/profile/m1/key/Parameter.StartingSeed HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.54.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Location: /api/model/modelOne/profile/m1/key/Parameter.StartingSeed
< Date: Fri, 28 Dec 2018 03:15:15 GMT
< Content-Length: 0
<
* Connection #0 to host localhost left intact
```

POST update workset read-only status

Update read-only status of model workset.

- Workset is a set of model input parameters (a.k.a. "scenario" input).
- Workset can be used to run the model.
- Workset is uniquely identified by name (in model context).
- Workset must be read-only in order to run the model with this set of input parameters.
- If user want to edit this set input parameters it must be read-write (not read-only status).

Methods:

```
POST /api/model/:model/workset/:set/readonly/:readonly
```

Arguments as URL parameters:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:set - (required) workset name
```

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default".

```
:readonly - (required) read-only status
```

Read-only status is a boolean value. It accepts 1, t, T, TRUE, true, True, 0, f, F, FALSE, false, False. Any other value returns an error.

Examples:

```
curl -v -X POST http://localhost:4040/api/model/modelOne/workset/modelOne_set/readonly/1

*   Trying ::1...
* TCP_NODELAY set
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 4040 (#0)
> POST /api/model/modelOne/workset/modelOne_set/readonly/1 HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.55.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Location: /api/model/_201208171604590148_/workset/modelOne_set
< Content-Type: application/json
< Date: Thu, 20 Dec 2018 03:54:59 GMT
< Content-Length: 122
<
{"SetId":3,"BaseRunId":0,"ModelId":1,"Name":"modelOne_set","IsReadonly":true,"UpdateDateTime":"2018-12-19 22:54:59.0583"}
* Connection #0 to host localhost left intact
```

```
curl -v -X POST http://localhost:4040/api/model/modelOne/workset/INVALID_NAME/readonly/1

*   Trying ::1...
* TCP_NODELAY set
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 4040 (#0)
> POST /api/model/modelOne/workset/INVALID_NAME/readonly/1 HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.55.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Date: Thu, 20 Dec 2018 03:56:50 GMT
< Content-Length: 87
<
{"SetId":0,"BaseRunId":0,"ModelId":0,"Name":"","IsReadonly":false,"UpdateDateTime":""}
* Connection #0 to host localhost left intact
```

PUT create new workset

Create new model workset and append new parameter(s) values.

- Workset is a set of model input parameters (a.k.a. "scenario" input).
- Workset can be used to run the model.
- Workset is uniquely identified by name (in model context).
- Workset must be read-only in order to run the model with this set of input parameters.
- If user want to edit this set input parameters it must be read-write (not read-only status).

This method creates new workset by inserting workset metadata, parameter(s) metadata and parameter(s) values from json request body. Workset metadata must contain model digest (or model name) and workset name, any other parts of metadata is optional.

Workset parameters are optional, you can create empty workset and add parameters later. Each parameter must have metadata and parameter value(s). Parameter metadata must contain parameter name and if cell values supplied then also number of sub-values (use 1 as default), all other metadata are optional. For each parameter values **must** be supplied. Workset cannot contain parameter metadata only, it must have all parameter values. See below for information about parameter values.

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default".

If workset with the same name already exist then method return an error.

If workset name not specified or empty then new workset created with unique name.

This is a beta version and may change in the future.

Method:

```
PUT /api/workset-create
```

For example:

```
curl -v -X PUT -H "Content-Type: application/json" "http://localhost:4040/api/workset-create" -d @test.json
```

JSON body:

It is expected to be same JSON metadata as return of Get Workset Metadata in All Languages method.

Parameter values must be one of:

- JSON cell values, identical to output of read parameter "page": Read parameter values from workset
- copy of parameter values from model run, use: `{ "Kind": "run", "From": "run digest or run name or run stamp" }`
- copy of parameter values from other workset, use: `{ "Kind": "set", "From": "workset name" }` Source workset must be `readonly`.

JSON response:

```
{
  "SetId": 142,
  "BaseRunId": 101,
  "ModelId": 1,
  "Name": "auto_name_set_of_parameters_2020_05_01_15_22_54_807",
  "IsReadonly": false,
  "UpdateDateTime": "2020-05-01 15:22:54.809"
}
```

Example 1:

```
{
  "ModelName": "modelOne",
  "Name": "NewSet",
  "Txt": [
    {
      "LangCode": "EN",
      "Descr": "New Set of model One parameters"
    }
  ]
}
```

Example 2:

```
{
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "NewSet",
  "BaseRunDigest": "",
  "IsReadonly": false,
  "Txt": [
    {
      "LangCode": "EN",
      "Descr": "New Set of model One parameters"
    },
    {
      "LangCode": "FR",
      "Descr": "(FR) New Set of model One parameters",
      "Note": "(FR) Note for New Set of model One parameters"
    }
  ],
  "Param": [
    {
      "Name": "ageSex",
      "SubCount": 1,
      "DefaultSubId": 0,
      "Txt": [
        {
          "LangCode": "EN",
          "Note": "Age by Sex new set of values"
        },
        {
          "LangCode": "FR",
          "Note": "(FR) Age by Sex new set of values"
        }
      ],
      "Value": [
        {"Dims": ["10-20","M"], "IsNull": false, "Value": 0.1, "SubId": 0},
        {"Dims": ["10-20","F"], "IsNull": false, "Value": 0.2, "SubId": 0},
        {"Dims": ["20-30","M"], "IsNull": false, "Value": 0.3, "SubId": 0},
        {"Dims": ["20-30","F"], "IsNull": false, "Value": 0.4, "SubId": 0},
        {"Dims": ["30-40","M"], "IsNull": false, "Value": 0.5, "SubId": 0},
        {"Dims": ["30-40","F"], "IsNull": false, "Value": 0.6, "SubId": 0},
        {"Dims": ["40+","M"],   "IsNull": false, "Value": 0.7, "SubId": 0},
        {"Dims": ["40+","F"],   "IsNull": false, "Value": 0.8, "SubId": 0}
      ]
    }
  ]
}
```

Example 3:

Create workset based on existing model run and copy two parameters:

- copy parameter `ageSex` from model run `Default-4`
- copy parameter `salaryAge` from `Default` workset

```
{
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "m1_based_on_first_run",
  "BaseRunDigest": "09cf2735bbe8aa88fd427c20925ca14a",
  "Txt": [{
      "LangCode": "EN",
      "Descr": "modelOne based on first run",
      "Note": "modelOne copy parameters from Default-4 run and Default workset"
    }, {
      "LangCode": "FR",
      "Descr": "(FR) modelOne based on first run",
      "Note": "(FR) modelOne copy parameters from Default-4 run and Default workset"
    }
  ],
  "Param": [
    {
      "Name": "ageSex",
      "Kind": "run",
      "From": "Default-4"
    },
    {
      "Name": "salaryAge",
      "Kind": "set",
      "From": "Default"
    }
  ]
}
```

```
curl -v -X PUT -H "Content-Type: application/json" "http://localhost:4040/api/workset-create" -d @m1_ws_new_on_run.json

*   Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 4040 (#0)
> PUT /api/workset-create HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.55.1
> Accept: */*
> Content-Type: application/json
> Content-Length: 399
>
* upload completely sent off: 399 out of 399 bytes
< HTTP/1.1 200 OK
< Content-Location: /api/model/_201208171604590148_/workset/auto_name_set_of_parameters_2020_05_01_15_22_54_807
< Content-Type: application/json
< Date: Fri, 01 May 2020 19:22:54 GMT
< Content-Length: 165
<
{"SetId":142,"BaseRunId":101,"ModelId":1,"Name":"auto_name_set_of_parameters_2020_05_01_15_22_54_807","IsReadonly":false,"UpdateDateTime":"2020-05-01 15:22:54.809"}
}
* Connection #0 to host localhost left intact
```

Example 4:

```
{
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "NewSet",
  "IsReadonly": true,
  "Txt": [
    {
      "LangCode": "EN",
      "Descr": "New Set of model One parameters"
    },
    {
      "LangCode": "FR",
      "Descr": "(FR) New Set of model One parameters",
      "Note": "(FR) Note for New Set of model One parameters"
    }
  ],
  "Param": [
    {
      "Name": "ageSex",
      "SubCount": 1,
      "DefaultSubId": 0,
      "Txt": [
        {
          "LangCode": "EN",
          "Note": "Age by Sex new set of values"
        },
        {
          "LangCode": "FR",
          "Note": "(FR) Age by Sex new set of values"
```

```
    }
  ],
  "Value": [
    {"Dims": ["10-20","M"], "IsNull": false, "Value": 0.1, "SubId": 0},
    {"Dims": ["10-20","F"], "IsNull": false, "Value": 0.2, "SubId": 0},
    {"Dims": ["20-30","M"], "IsNull": false, "Value": 0.3, "SubId": 0},
    {"Dims": ["20-30","F"], "IsNull": false, "Value": 0.4, "SubId": 0},
    {"Dims": ["30-40","M"], "IsNull": false, "Value": 0.5, "SubId": 0},
    {"Dims": ["30-40","F"], "IsNull": false, "Value": 0.6, "SubId": 0},
    {"Dims": ["40+","M"],   "IsNull": false, "Value": 0.7, "SubId": 0},
    {"Dims": ["40+","F"],   "IsNull": false, "Value": 0.8, "SubId": 0}
  ]
 },
 {
  "Name": "salaryAge",
  "SubCount": 1,
  "DefaultSubId": 0,
  "Txt": [
    {
      "LangCode": "EN",
      "Note": "Salary by Age new set of values"
    },
    {
      "LangCode": "FR",
      "Note": "(FR) Salary by Age new set of values"
    }
  ],
  "Value": [
    {"Dims": ["L","10-20"], "IsNull": false, "Value": 10, "SubId": 0},
    {"Dims": ["L","20-30"], "IsNull": false, "Value": 20, "SubId": 0},
    {"Dims": ["L","30-40"], "IsNull": false, "Value": 30, "SubId": 0},
    {"Dims": ["L","40+"],   "IsNull": false, "Value": 40, "SubId": 0},
    {"Dims": ["M","10-20"], "IsNull": false, "Value": 11, "SubId": 0},
    {"Dims": ["M","20-30"], "IsNull": false, "Value": 21, "SubId": 0},
    {"Dims": ["M","30-40"], "IsNull": false, "Value": 31, "SubId": 0},
    {"Dims": ["M","40+"],   "IsNull": false, "Value": 41, "SubId": 0},
    {"Dims": ["H","10-20"], "IsNull": false, "Value": 12, "SubId": 0},
    {"Dims": ["H","20-30"], "IsNull": false, "Value": 22, "SubId": 0},
    {"Dims": ["H","30-40"], "IsNull": false, "Value": 32, "SubId": 0},
    {"Dims": ["H","40+"],   "IsNull": false, "Value": 42, "SubId": 0}
  ]
 },
 {
  "Name": "StartingSeed",
  "SubCount": 1,
  "DefaultSubId": 0,
  "Txt": [
    {
      "LangCode": "EN",
      "Note": "Starting seed new set of value"
    }
  ],
  "Value": [
    {"Dims": [], "IsNull": false, "Value": 8191, "SubId": 0}
  ]
 },
 {
  "Name": "salaryFull",
  "SubCount": 4,
  "DefaultSubId": 3,
  "Txt": [
    {
      "LangCode": "EN",
      "Note": "Full or part time by Salary new set of values"
    }
  ],
  "Value": [
    {"Dims": ["L"], "IsNull": false, "Value": "Part", "SubId": 0},
    {"Dims": ["M"], "IsNull": false, "Value": "Full", "SubId": 0},
    {"Dims": ["H"], "IsNull": false, "Value": "Full", "SubId": 0},
    {"Dims": ["L"], "IsNull": false, "Value": "Part", "SubId": 1},
    {"Dims": ["M"], "IsNull": false, "Value": "Part", "SubId": 1},
    {"Dims": ["H"], "IsNull": false, "Value": "Part", "SubId": 1},
    {"Dims": ["L"], "IsNull": false, "Value": "Full", "SubId": 2},
    {"Dims": ["M"], "IsNull": false, "Value": "Full", "SubId": 2},
    {"Dims": ["H"], "IsNull": false, "Value": "Full", "SubId": 2},
    {"Dims": ["L"], "IsNull": false, "Value": "Full", "SubId": 3},
    {"Dims": ["M"], "IsNull": false, "Value": "Full", "SubId": 3},
    {"Dims": ["H"], "IsNull": false, "Value": "Part", "SubId": 3}
  ]
 },
 {
  "Name": "baseSalary",
  "SubCount": 4,
  "DefaultSubId": 3,
  "Txt": [],
  "Value": [
    {"Dims": [],"IsNull": false, "Value": "Full", "SubId": 0},
```

```
      {"Dims": [],"IsNull": false, "Value": "Part", "SubId": 1},
      {"Dims": [],"IsNull": false, "Value": "Full", "SubId": 2},
      {"Dims": [],"IsNull": false, "Value": "Part", "SubId": 3}
    ]
  },
  {
    "Name": "filePath",
    "SubCount": 4,
    "DefaultSubId": 3,
    "Txt": [],
    "Value": [
      {"Dims": [], "IsNull": false, "Value": "file 0 path", "SubId": 0},
      {"Dims": [], "IsNull": false, "Value": "file 1 path", "SubId": 1},
      {"Dims": [], "IsNull": false, "Value": "file 2 path", "SubId": 2},
      {"Dims": [], "IsNull": false, "Value": "file 3 path", "SubId": 3}
    ]
  },
  {
    "Name": "isOldAge",
    "SubCount": 4,
    "DefaultSubId": 3,
    "Txt": [
      {
        "LangCode": "EN",
        "Note": "Is old age new set of values"
      }
    ],
    "Value": [
      {"Dims": ["10-20"], "IsNull": false, "Value": false, "SubId": 0},
      {"Dims": ["20-30"], "IsNull": false, "Value": false, "SubId": 0},
      {"Dims": ["30-40"], "IsNull": false, "Value": false, "SubId": 0},
      {"Dims": ["40+"],   "IsNull": false, "Value": true,  "SubId": 0},
      {"Dims": ["10-20"], "IsNull": false, "Value": false, "SubId": 1},
      {"Dims": ["20-30"], "IsNull": false, "Value": false, "SubId": 1},
      {"Dims": ["30-40"], "IsNull": false, "Value": false, "SubId": 1},
      {"Dims": ["40+"],   "IsNull": false, "Value": true,  "SubId": 1},
      {"Dims": ["10-20"], "IsNull": false, "Value": false, "SubId": 2},
      {"Dims": ["20-30"], "IsNull": false, "Value": false, "SubId": 2},
      {"Dims": ["30-40"], "IsNull": false, "Value": false, "SubId": 2},
      {"Dims": ["40+"],   "IsNull": false, "Value": true,  "SubId": 2},
      {"Dims": ["10-20"], "IsNull": false, "Value": false, "SubId": 3},
      {"Dims": ["20-30"], "IsNull": false, "Value": false, "SubId": 3},
      {"Dims": ["30-40"], "IsNull": false, "Value": false, "SubId": 3},
      {"Dims": ["40+"],   "IsNull": false, "Value": true,  "SubId": 3}
    ]
  }
 ]
}
```

PUT create or replace workset

Create new or replace existing model workset metadata, parameter(s) metadata and parameter(s) values.

- Workset is a set of model input parameters (a.k.a. "scenario" input).
- Workset can be used to run the model.
- Workset is uniquely identified by name (in model context).
- Workset must be read-only in order to run the model with this set of input parameters.
- If user want to edit this set input parameters it must be read-write (not read-only status).

This method replace workset metadata, parameter(s) metadata and parameter(s) values from multipart-form, expected multipart form parts:

- first workset part with workset metadata and parameters metadata in json
- optional multiple parts file-csv=parameterName.csv.

If no such workset exist in database then new workset created.

If workset name not specified or empty then new workset created with unique name.

If workset already exist then it is delete-insert operation:

- existing metadata, parameter list, parameter metadata and parameter values deleted from database;
- new metadata, parameters metadata and parameters values inserted.

For each parameter in the parameter list csv parameter values **must** be supplied. Workset cannot contain parameter metadata only, it must have parameter values as parameter.csv part.

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default".

This is a beta version and may change in the future.

Method:

```
PUT /api/workset-replace
```

For example:

```
curl -v -X PUT -F "workset=@test.json" http://localhost:4040/api/workset-replace
curl -v -X PUT -F "workset=@test.json" -F "parameter-csv=@new_ageSex.csv;filename=ageSex.csv" http://localhost:4040/api/workset-replace
```

JSON arguments:

It is expected to be same JSON as return of Get Workset Metadata in All Languages method.

For example (test.json file):

```
{
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "modelOne_set2",
  "BaseRunDigest": "",
  "IsReadonly": false,
  "Txt": [{
      "LangCode": "EN",
      "Descr": "modelOne modified set of parameters",
      "Note": ""
    }, {
      "LangCode": "FR",
      "Descr": "(FR) modelOne modified set of parameters",
      "Note": "(FR) modelOne workset notes"
    }
  ],
  "Param": [{
      "Name": "ageSex",
      "SubCount": 1,
      "DefaultSubId": 0,
      "Txt": [{
          "LangCode": "EN",
          "Note": "Age by Sex modified values"
        }, {
          "LangCode": "FR",
          "Note": "(FR) Age by Sex modified values"
        }
      ]
    }
  ]
}
```


Each parameter.csv part expected to be same as return of methods:

- GET parameter values from model run
- GET parameter values from workset

For example (new_ageSex.csv file):

```
sub_id,dim0,dim1,param_value
0,10-20,M,1.1
0,10-20,F,1.2
0,20-30,M,1.3
0,20-30,F,1.4
0,30-40,M,1.5
0,30-40,F,1.6
0,40+,M,1.7
0,40+,F,1.8
```

JSON response:


```
{
  "SetId": 142,
  "BaseRunId": 101,
  "ModelId": 1,
  "Name": "auto_name_set_of_parameters_2020_05_01_15_22_54_807",
  "IsReadonly": false,
  "UpdateDateTime": "2020-05-01 15:22:54.809"
}
```

PATCH create or merge workset

Create new or merge existing model workset metadata, parameter(s) metadata and replace parameter(s) values.

- Workset is a set of model input parameters (a.k.a. "scenario" input).
- Workset can be used to run the model.
- Workset is uniquely identified by name (in model context).
- Workset must be read-only in order to run the model with this set of input parameters.
- If user want to edit this set input parameters it must be read-write (not read-only status).

This method merge workset metadata, parameter(s) metadata and parameter(s) values from multipart-form, expected multipart form parts:

- first workset part with workset metadata and parameters metadata in json
- optional multiple parts file-csv=parameterName.csv.

First part must have model digest or name and workset name:

- if no such workset exist in database then new workset created.
- if workset already exist then merge existing workset metadata with new.
- if workset name not specified or empty then new workset created with unique name.

Parameter list merged with existing workset parameter list:

- if parameter exist in workset then parameter metadata merged.
- if new parameter values supplied then replace parameter values.
- if parameter not already exist in workset then parameter values **must** be supplied.

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default".

This is a beta version and may change in the future.

Method:

```
PATCH /api/workset-merge
```

For example:

```
curl -v -X PATCH -F "workset=@test.json" http://localhost:4040/api/workset-merge
curl -v -X PATCH -F "workset=@test.json" -F "parameter-csv=@new_ageSex.csv;filename=ageSex.csv" http://localhost:4040/api/workset-merge
```

JSON arguments:

It is expected to be same JSON as return of Get Workset Metadata in All Languages method.

For example (test.json file):

```
{
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "modelOne_set2",
  "BaseRunDigest": "",
  "IsCleanBaseRun": true,
  "IsReadonly": false,
  "Txt": [{
      "LangCode": "EN",
      "Descr": "modelOne modified set of parameters",
      "Note": ""
    }, {
      "LangCode": "FR",
      "Descr": "(FR) modelOne modified set of parameters",
      "Note": "(FR) modelOne workset notes"
    }
  ],
  "Param": [{
      "Name": "ageSex",
      "SubCount": 1,
      "Txt": [{
          "LangCode": "EN",
          "Note": "Age by Sex modified values"
        }, {
          "LangCode": "FR",
          "Note": "(FR) Age by Sex modified values"
        }
      ]
    }
  ]
}
```

Use `"IsCleanBaseRun": true` and `"BaseRunDigest": ""` if you want to update base run to empty `NULL` value. Use only `"BaseRunDigest": ""` if you do not want to update base run.

Each parameter.csv part expected to be same as return of methods:

- GET parameter values from model run
- GET parameter values from workset

For example (new_ageSex.csv file):

```
sub_id,dim0,dim1,param_value
0,10-20,M,1.1
0,10-20,F,1.2
0,20-30,M,1.3
0,20-30,F,1.4
0,30-40,M,1.5
0,30-40,F,1.6
0,40+,M,1.7
0,40+,F,1.8
```

JSON response:

```
{
  "SetId": 142,
  "BaseRunId": 101,
  "ModelId": 1,
  "Name": "auto_name_set_of_parameters_2020_05_01_15_22_54_807",
  "IsReadonly": false,
  "UpdateDateTime": "2020-05-01 15:22:54.809"
}
```

DELETE workset

Delete model workset and workset parameter(s) values from database.

Workset is a set of model input parameters (a.k.a. "scenario" input). Workset can be used to run the model.

Method:

```
DELETE /api/model/:model/workset/:set
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:set - (required) workset name
```

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default".

Call examples:

```
curl -v -X DELETE http://localhost:4040/api/model/modelOne/workset/modelOne_set2
curl -v -X DELETE http://localhost:4040/api/model/_201208171604590148_/workset/modelOne_set2
```

```
curl -v -X DELETE http://localhost:4040/api/model/modelOne/workset/modelOne_set2

*   Trying ::1...
* TCP_NODELAY set
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 4040 (#0)
> DELETE /api/model/modelOne/workset/modelOne_set2 HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.54.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Location: /api/model/modelOne/workset/modelOne_set2
< Date: Tue, 19 Dec 2017 03:10:16 GMT
< Content-Length: 0
< Content-Type: text/plain; charset=utf-8
<
* Connection #0 to host localhost left intact
```

DELETE parameter from workset

Delete parameter from workset: delete workset parameter metadata and parameter valuesfrom database.

Workset is a set of model input parameters (a.k.a. "scenario" input). Workset can be used to run the model.

Method:

```
DELETE /api/model/:model/workset/:set/parameter/:name
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:set - (required) workset name
```

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default".

```
:name - (required) parameter name
```

Call examples:

```
curl -v -X DELETE http://localhost:4040/api/model/modelOne/workset/modelOne_set2/parameter/ageSex
curl -v -X DELETE http://localhost:4040/api/model/_201208171604590148_/workset/modelOne_set2/parameter/ageSex
```

```
curl -v -X DELETE http://localhost:4040/api/model/modelOne/workset/modelOne_set2/parameter/ageSex

*   Trying ::1...
* TCP_NODELAY set
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 4040 (#0)
> DELETE /api/model/modelOne/workset/modelOne_set2/parameter/ageSex HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.54.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Location: /api/model/modelOne/workset/modelOne_set2/parameter/ageSex
< Date: Fri, 22 Dec 2017 03:16:54 GMT
< Content-Length: 0
< Content-Type: text/plain; charset=utf-8
<
* Connection #0 to host localhost left intact
```

PATCH update workset parameter values

Update existing parameter values in workset.

Workset is a set of model input parameters (a.k.a. "scenario" input). Workset can be used to run the model.

This method replace existing parameter values by new values. Typical use of this method is a parameter editing through UI when only small part of parameter rows replaced. Input data are json-encoded and expected to be same as `Page` part of JSON return from Read parameter values from workset method.

Dimension(s) and enum-based parameter values expected as enum codes.

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default".

This is a beta version and may change in the future.

Method:

```
PATCH /api/model/:model/workset/:set/parameter/:name/new/value
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:set - (required) workset name
```

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default".

```
:name - (required) parameter name
```

Call examples:

```
curl -v -X PATCH -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/workset/modelOne_set/parameter/ageSex/new/value -d @test.json
curl -v -X PATCH -H "Content-Type: application/json" http://localhost:4040/api/model/_201208171604590148_/workset/modelOne_set/parameter/ageSex/new/value -d @test.json
```

JSON body:

It is expected to be same as `Page` part of JSON return from Read parameter values from workset method.

For example (test.json file):

```
[
{"Dims":["10-20","M"],"IsNull":false,"Value":1234.1,"SubId":0}
,{"Dims":["10-20","F"],"IsNull":false,"Value":5678.2,"SubId":0}
]
```

PATCH update workset parameter values (enum id's)

Update existing parameter values in workset.

Workset is a set of model input parameters (a.k.a. "scenario" input). Workset can be used to run the model.

This method replace existing parameter values by new values. Typical use of this method is a parameter editing through UI when only small part of parameter rows replaced. Input data are json-encoded and expected to be the same as `Page` part of JSON return from Read parameter values from workset (enum id's) method.

Dimension(s) and enum-based parameter values expected as enum id, not enum codes.

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default".

This is a beta version and may change in the future.

Method:

```
PATCH /api/model/:model/workset/:set/parameter/:name/new/value-id
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:set - (required) workset name
```

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default".

```
:name - (required) parameter name
```

Call examples:

```
curl -v -X PATCH -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/workset/modelOne_set/parameter/ageSex/new/value-id -d @test.json
curl -v -X PATCH -H "Content-Type: application/json" http://localhost:4040/api/model/_201208171604590148_/workset/modelOne_set/parameter/ageSex/new/value-id -d @test.json
```

JSON body:

It is expected to be same as `Page` part of JSON return from Read parameter values from workset (enum id's) method.

For example (test.json file):

```
[
{"DimIds":[10,0],"IsNull":false,"Value":9876.1,"SubId":0}
,{"DimIds":[10,1],"IsNull":false,"Value":5432.2,"SubId":0}
]
```

PATCH update workset parameter(s) value notes

Update parameter(s) value notes in workset.

Workset is a set of model input parameters (a.k.a. "scenario" input). Workset can be used to run the model.

This method merge (insert new or update existing) parameter(s) value notes in workset. Input data are json-encoded array of parameters, similat to the `Param` array part of JSON return from GET workset including text in all languages method.

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model.

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default".

Workset must be in read-write state (editable) otherwise error returned.

Each input element of parameters value notes array must have parameter `Name` and can have optional `Txt` array with `LangCode` and `Note` properties for each element. For example:

```
[
  {
    "Name": "ageSex",
    "Txt": [
      {
        "LangCode": "EN",
        "Note": "Age by Sex Default values"
      },
      {
        "LangCode": "FR",
        "Note": "Valeurs par défaut de l'âge par sexe"
      }
    ]
  },
  {
    "Name": "salaryAge",
    "Txt": [
      {
        "LangCode": "EN",
        "Note": "Salary by Age default values"
      },
      {
        "LangCode": "FR",
        "Note": "Salaire par âge valeurs par défaut"
      }
    ]
  }
]
```

Parameter `Name` must be one of the name of workset parameter (parameter already included in workset). If parameter does not included in workset then error returned.

`LangCode` must be one of model alguage codes or dialect of such, for example, it can be `fr-CA` or `fr-FR` if model has `FR` language. `Note` value can be empty.

If parameter value notes already exist for such parameter `Name` and `LngCode` then it will be replaced with new `Note` value. If there is no such parameter note then new value will be inserted.

This is a beta version and may change in the future.

Method:

```
PATCH /api/model/:model/workset/:set/parameter-text
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:set - (required) workset name
```

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default".

JSON arguments:

It is expected to be same JSON as return of Get Workset Metadata in All Languages method.

For example (test.json file):


```
[{
    "Name": "ageSex",
    "Txt": [{
        "LangCode": "EN",
        "Note": "new value notes Age by Sex"
      }, {
        "LangCode": "FR",
        "Note": "nouvelles notes de valeur Âge par sexe"
      }
    ]
  }, {
    "Name": "StartingSeed",
    "Txt": [{
        "LangCode": "EN",
        "Note": "new value notes Starting seed"
      }
    ]
  }, {
    "Name": "baseSalary",
    "Txt": [{
        "LangCode": "EN",
        "Note": "new value notes Base Salary"
      }
    ]
  }
]
```


Call examples:

```
curl -v -X PATCH -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/workset/Default/parameter-text -d @test.json
curl -v -X PATCH -H "Content-Type: application/json" http://localhost:4040/api/model/_201208171604590148_/workset/Default/parameter-text -d @test.json
```

Output example:

```
* Connected to localhost (::1) port 4040 (#0)
> PATCH /api/model/_201208171604590148_/workset/Default/parameter-text HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.55.1
> Accept: */*
> Content-Type: application/json
> Content-Length: 469
>
* upload completely sent off: 469 out of 469 bytes
< HTTP/1.1 200 OK
< Content-Location: /api/model/_201208171604590148_/workset/Default/parameter-text
< Content-Type: text/plain
< Date: Tue, 26 Oct 2021 02:34:21 GMT
< Content-Length: 0
<
* Connection #0 to host localhost left intact
```

PUT copy parameter from model run into workset

Copy and insert new parameter values and parameter value notes from model run into workset.

If parameter with that name already exist in workset then error returned. There is similar method PATCH merge parameter from model run into workset which does insert new or replace existing parameter.

- Workset is a set of model input parameters (a.k.a. "scenario" input).
- Workset can be used to run the model.
- Workset must be read-only in order to run the model with this set of input parameters.
- If user want to edit this set input parameters it must be read-write (not read-only status).

Method:

```
PUT /api/model/:model/workset/:set/copy/parameter/:name/from-run/:run
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:set - (required) workset name
```

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default". Destination workset must be in read-write state (editable), use POST update workset read-only status method to make workset editable.

```
:name - (required) parameter name
```

If parameter with that name already exist in workset then error returned. You can delete parameter from workset by DELETE parameter from workset method.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

Call examples:

```
curl -v -X PUT http://localhost:4040/api/model/modelOne/workset/set2/copy/parameter/ageSex/from-run/Default-4
curl -v -X PUT http://localhost:4040/api/model/_201208171604590148_/workset/set2/copy/parameter/ageSex/from-run/6fbad822cb9ae42deea1ede626890711
curl -v -X PUT http://localhost:4040/api/model/modelOne/workset/set2/copy/parameter/ageSex/from-run/2019_01_17_19_59_52_998
```

```
curl -v -X PUT http://localhost:4040/api/model/modelOne/workset/set2/copy/parameter/ageSex/from-run/Default-4

*   Trying ::1...
* TCP_NODELAY set
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 4040 (#0)
> PUT /api/model/modelOne/workset/set2/copy/parameter/ageSex/from-run/Default-4 HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.54.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Location: /api/model/modelOne/workset/set2/parameter/ageSex
< Date: Mon, 31 Dec 2018 19:34:21 GMT
< Content-Length: 0
<
* Connection #0 to host localhost left intact
```

PATCH merge parameter from model run into workset

Copy and insert or replace parameter values and parameter value notes from model run into workset.

If parameter with that name already exist in workset then existing values and parameter metadata replaced by new copy from model run. There is similar method PUT copy parameter from model run into workset which returns error if parameter already exist in workset.

- Workset is a set of model input parameters (a.k.a. "scenario" input).
- Workset can be used to run the model.
- Workset must be read-only in order to run the model with this set of input parameters.
- If user want to edit this set input parameters it must be read-write (not read-only status).

Method:

```
PATCH /api/model/:model/workset/:set/merge/parameter/:name/from-run/:run
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:set - (required) workset name
```

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default". Destination workset must be in read-write state (editable), use POST update workset read-only status method to make workset editable.

```
:name - (required) parameter name
```

If parameter with that name already exist in workset then it is delete and insert operation: existing parameter values and metadata will be replaced by copy from model run.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

Call examples:

```
curl -v -X PATCH http://localhost:4040/api/model/modelOne/workset/NewSet/merge/parameter/salaryFull/from-run/Default-4
curl -v -X PATCH http://localhost:4040/api/model/_201208171604590148_/workset/NewSet/merge/parameter/salaryFull/from-run/3356660729aaaaccf04f1699248c4355
curl -v -X PATCH http://localhost:4040/api/model/modelOne/workset/NewSet/merge/parameter/salaryFull/from-run/2021_10_04_21_19_18_975
```

```
curl -v -X PATCH http://localhost:4040/api/model/modelOne/workset/NewSet/merge/parameter/salaryFull/from-run/Default-4

*   Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 4040 (#0)
> PATCH /api/model/modelOne/workset/NewSet/merge/parameter/salaryFull/from-run/Default-4 HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.55.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Location: /api/model/modelOne/workset/NewSet/parameter/salaryFull
< Content-Type: text/plain
< Date: Sat, 30 Oct 2021 03:36:11 GMT
< Content-Length: 0
<
* Connection #0 to host localhost left intact
```

PUT copy parameter from workset to another

Copy and insert new parameter values and parameter value notes from one workset to another.

If parameter with that name already exist in workset then error returned. There is similar method PATCH merge parameter from workset to another which does insert new or replace existing parameter.

- Workset is a set of model input parameters (a.k.a. "scenario" input).
- Workset can be used to run the model.
- Workset must be read-only in order to run the model with this set of input parameters.
- If user want to edit this set input parameters it must be read-write (not read-only status).

Method:

```
PUT /api/model/:model/workset/:set/copy/parameter/:name/from-workset/:from-set
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:set - (required) destination workset name
```

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default". Destination workset must be in read-write state (editable), use POST update workset read-only status method to make workset editable.

```
:name - (required) parameter name
```

If parameter with that name already exist in workset then error returned. You can delete parameter from workset by DELETE parameter from workset method.

```
:from-set - (required) source workset name
```

Source workset must be in read-only state.

Call examples:

```
curl -v -X PUT http://localhost:4040/api/model/modelOne/workset/set2/copy/parameter/ageSex/from-workset/modelOne_other
curl -v -X PUT http://localhost:4040/api/model/_201208171604590148_/workset/set2/copy/parameter/ageSex/from-workset/modelOne_other
```

```
curl -v -X PUT http://localhost:4040/api/model/modelOne/workset/set2/copy/parameter/ageSex/from-workset/modelOne_other

*   Trying ::1...
* TCP_NODELAY set
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 4040 (#0)
> PUT /api/model/modelOne/workset/set2/copy/parameter/ageSex/from-workset/modelOne_other HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.54.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Location: /api/model/modelOne/workset/set2/parameter/ageSex
< Date: Mon, 31 Dec 2018 19:50:05 GMT
< Content-Length: 0
<
* Connection #0 to host localhost left intact
```

PATCH merge parameter from workset to another

Copy and insert or replace parameter values and parameter value notes from one workset to another.

If parameter with that name already exist in workset then existing values and parameter metadata replaced by new copy from source workset. There is similar method PUT copy parameter from workset to another which returns error if parameter already exist in workset.

- Workset is a set of model input parameters (a.k.a. "scenario" input).
- Workset can be used to run the model.
- Workset must be read-only in order to run the model with this set of input parameters.
- If user want to edit this set input parameters it must be read-write (not read-only status).

Method:

```
PATCH /api/model/:model/workset/:set/merge/parameter/:name/from-workset/:from-set
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:set - (required) destination workset name
```

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default". Destination workset must be in read-write state (editable), use POST update workset read-only status method to make workset editable.

```
:name - (required) parameter name
```

If parameter with that name already exist in workset then it is delete and insert operation: existing parameter values and metadata will be replaced by copy from source workset.

```
:from-set - (required) source workset name
```

Source workset must be in read-only state.

Call examples:

```
curl -v -X PATCH http://localhost:4040/api/model/modelOne/workset/NewSet/merge/parameter/salaryFull/from-workset/Default

* Connected to localhost (::1) port 4040 (#0)
> PATCH /api/model/modelOne/workset/NewSet/merge/parameter/salaryFull/from-workset/Default HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.55.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Location: /api/model/modelOne/workset/NewSet/parameter/salaryFull
< Content-Type: text/plain
< Date: Sat, 30 Oct 2021 02:59:59 GMT
< Content-Length: 0
<
* Connection #0 to host localhost left intact
```

PATCH update model run text (description and notes)

Merge (add new or update existing) model run text (description and notes and run parameters value notes.

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

Model run must be completed (successfully or with error) before you can modify run text description or notes. If model run still in progress then error returned.

This is a beta version and may change in the future.

Method:

```
PATCH /api/run/text
```

For example:

```
curl -v -X PATCH -H "Content-Type: application/json" http://localhost:4040/api/run/text -d @test.json
```

JSON arguments:

It is expected to be same JSON as return of GET run including text in all languages method.

Only following parts are used from input json:

- model digest or model name
- run digest, run stamp or run name
- run text language code, description and notes
- parameter value text language code and notes Any other parts on json body are silently ignored because it is not possible to modify model run data, only run text (dexcription and notes) can be updated.

For example (test.json file):

```
{
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "Default-4",
  "Digest": "05403de52f30f59b050417561914fbb8",
  "Txt": [
    {
      "LangCode": "EN",
      "Descr": "UPDATED Model One default set of parameters",
      "Note": "UPDATED Note"
    }
  ],
  "Param": [
    {
      "Name": "ageSex",
      "Txt": [
        {
          "LangCode": "EN",
          "Note": "UPDATED Age by Sex default values"
        }
      ]
    },
    {
      "Name": "salaryAge",
      "Txt": [
        {
          "LangCode": "EN",
          "Note": "UPDATED Salary by Age default values"
        }
      ]
    }
  ]
}
```

DELETE model run

Delete model run results from database, including output table values and values of run input parameters.

Method:

```
DELETE /api/model/:model/run/:run
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

Call examples:

```
curl -v -X DELETE http://localhost:4040/api/model/modelOne/run/Default-4
curl -v -X DELETE http://localhost:4040/api/model/_201208171604590148_/run/05403de52f30f59b050417561914fbb8
curl -v -X DELETE http://localhost:4040/api/model/modelOne/run/2019_01_17_19_59_52_998
```

```
curl -v -X DELETE http://localhost:4040/api/model/modelOne/run/Default-4
*   Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 4040 (#0)
> DELETE /api/model/modelOne/run/Default-4 HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.54.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Location: /api/model/modelOne/run/Default-4
< Date: Fri, 11 Jan 2019 02:25:48 GMT
< Content-Length: 0
<
* Connection #0 to host localhost left intact
```

PATCH update run parameter(s) value notes

Update parameter(s) value notes in model run.

This method merge (insert new or update existing) parameter(s) value notes in model run. Input data are json-encoded array of parameters, similat to the `Param` array part of JSON return from GET model run including text in all languages method.

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model.

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run.

Each input element of parameters value notes array must have parameter `Name` and can have optional `Txt` array with `LangCode` and `Note` properties for each element. For example:

```
[
  {
    "Name": "ageSex",
    "Txt": [
      {
        "LangCode": "EN",
        "Note": "Age by Sex Default values"
      },
      {
        "LangCode": "FR",
        "Note": "Valeurs par défaut de l'âge par sexe"
      }
    ]
  },
  {
    "Name": "salaryAge",
    "Txt": [
      {
        "LangCode": "EN",
        "Note": "Salary by Age default values"
      },
      {
        "LangCode": "FR",
        "Note": "Salaire par âge valeurs par défaut"
      }
    ]
  }
]
```

`LangCode` must be one of model alguage codes or dialect of such, for example, it can be `fr-CA` or `fr-FR` if model has `FR` language. `Note` value can be empty.

If parameter value notes already exist for such parameter `Name` and `LngCode` then it will be replaced with new `Note` value. If there is no such parameter note then new value will be inserted.

This is a beta version and may change in the future.

Method:

```
PATCH /api/model/:model/run/:run/parameter-text
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

JSON arguments:

It is expected to be same JSON as return of GET model run including text in all languages method.

For example (test.json file):

```
[{
    "Name": "ageSex",
    "Txt": [{
        "LangCode": "EN",
        "Note": "new value notes Age by Sex"
      }, {
        "LangCode": "FR",
        "Note": "nouvelles notes de valeur Âge par sexe"
      }
    ]
  }, {
    "Name": "StartingSeed",
    "Txt": [{
        "LangCode": "EN",
        "Note": "new value notes Starting seed"
      }
    ]
  }, {
    "Name": "baseSalary",
    "Txt": [{
        "LangCode": "EN",
        "Note": "new value notes Base Salary"
      }
    ]
  }
]
```

Call examples:

```
curl -v -X PATCH -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/run/Default-4/parameter-text -d @test.json
curl -v -X PATCH -H "Content-Type: application/json" http://localhost:4040/api/model/_201208171604590148_/run/3356660729aaaaccf04f1699248c4355/parameter-text -d @test.json
```

Output example:

```
* Connected to localhost (::1) port 4040 (#0)
> PATCH /api/model/_201208171604590148_/run/3356660729aaaaccf04f1699248c4355/parameter-text HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.55.1
> Accept: */*
> Content-Type: application/json
> Content-Length: 469
>
* upload completely sent off: 469 out of 469 bytes
< HTTP/1.1 200 OK
< Content-Location: /api/model/_201208171604590148_/run/3356660729aaaaccf04f1699248c4355/parameter-text
< Content-Type: text/plain
< Date: Tue, 26 Oct 2021 02:48:34 GMT
< Content-Length: 0
<
* Connection #0 to host localhost left intact
```

PUT create or replace modeling task

Create new or replace existing modeling task definition: including task text (description and notes) and list of task input sets (worksets).

It does delete existing and insert new rows into task_txt and task_set db tables. If task does not exist then new task created by inserting into task_lst table.

Following parts can be submitted as JSON body (see example below):

- model name
- model digest
- task name
- task text as array of: language code, description, notes
- task input worksets as array of workset names

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

Task is uniquely identified by name (inside the model). Different models can have tasks with same name, i.e. each model can have task with name "My First Task".

If task name not specified or empty then new task created with unique name.

Task input worksets must already exist in database: all workset names must exist in workset_lst table.

This is a beta version and may change in the future.

Method:

```
PUT  /api/task-new
```

For example:

```
curl -v -X PUT  -H "Content-Type: application/json" http://localhost:4040/api/task-new -d @test.json
```

JSON argument:

It is expected to be similar JSON return of GET task including text in all languages method. It can include only following parts of GET results:

- Task.ModelName
- Task.ModelDigest
- Task.Name
- Task.Txt
- Task.Set

For example (test.json file):

```
{
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "task-2",
  "Txt": [{
      "LangCode": "EN",
      "Descr": "Task Two for Model One",
      "Note": "Task Two: two set of input parameters"
    }, {
      "LangCode": "FR",
      "Descr": "(FR) Task Two for Model One",
      "Note": ""
    }
  ],
  "Set": [
    "modelOne_other"
  ]
}
```

JSON response:

```
{
  "Name":"auto_name_task_2020_05_01_15_25_38_208"
}
```

PATCH create or update modeling task

Create new or merge existing modeling task definition: including task text (description and notes) and list of task input sets (worksets).

It does update existing or insert new rows into task_txt and task_set db tables. If task does not exist then new task created by inserting into task_lst table.

Following parts can be submitted as JSON body (see example below):

- model name
- model digest
- task name
- task text as array of: language code, description, notes
- task input worksets as array of workset names

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

Task is uniquely identified by name (inside the model). Different models can have tasks with same name, i.e. each model can have task with name "My First Task".

If task name not specified or empty then new task created with unique name.

Task input worksets must already exist in database: all workset names must exist in workset_lst table.

This is a beta version and may change in the future.

Method:

```
PATCH /api/task
```

For example:

```
curl -v -X PATCH -H "Content-Type: application/json" http://localhost:4040/api/task -d @test.json
```

JSON argument:

It is expected to be similar JSON return of GET task including text in all languages method. It can include only following parts of GET results:

- Task.ModelName
- Task.ModelDigest
- Task.Name
- Task.Txt
- Task.Set

For example (test.json file):

```
{
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "Name": "task-2",
  "Txt": [{
      "LangCode": "EN",
      "Descr": "UPDATED Task Two for Model One",
      "Note": "UPDATED Task Two: two set of input parameters"
    }, {
      "LangCode": "FR",
      "Descr": "(FR) Task Two for Model One",
      "Note": "UPDATED notes"
    }
  ],
  "Set": [
    "Default"
  ]
}
```

JSON response:

```
{
  "Name":"auto_name_task_2020_05_01_15_25_38_208"
}
```

Example:

```
curl -v -X PATCH -H "Content-Type: application/json" http://localhost:4040/api/task -d @task_t2_def_merge.json

*   Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 4040 (#0)
> PATCH /api/task HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.55.1
> Accept: */*
> Content-Type: application/json
> Content-Length: 364
>
* upload completely sent off: 364 out of 364 bytes
< HTTP/1.1 200 OK
< Content-Location: /api/model/_201208171604590148_/task/auto_name_task_2020_05_01_15_25_38_208
< Content-Type: application/json
< Date: Fri, 01 May 2020 19:25:38 GMT
< Content-Length: 50
<
{"Name":"auto_name_task_2020_05_01_15_25_38_208"}
* Connection #0 to host localhost left intact
```

DELETE modeling task

Delete modeling task and task run history from database.

Model run results are not deleted and model input parameter values are not deleted. Only task and task run history deleted. Delete done only from task_lst, task_txt, task_set, task_run_lst and task_run_set db tables.

Method:

```
DELETE /api/model/:model/task/:task
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:task - (required) modeling task name
```

Task is uniquely identified by name (inside the model). Different models can have tasks with same name, i.e. each model can have task with name "My First Task".

Call examples:

```
curl -v -X DELETE http://localhost:4040/api/model/modelOne/task/task-2
curl -v -X DELETE http://localhost:4040/api/model/_201208171604590148_/task/task-2
```

```
curl -v -X DELETE http://localhost:4040/api/model/modelOne/task/task-2
*   Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 4040 (#0)
> DELETE /api/model/modelOne/task/task-2 HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.54.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Location: /api/model/modelOne/task/task-2
< Date: Sat, 12 Jan 2019 00:50:07 GMT
< Content-Length: 0
<
* Connection #0 to host anatolyv17om left intact
```

POST a request to run the model

Start new model run.

This is a beta version and may change in the future.

Method:

```
POST /api/run
```

For example:

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/run -d @run_modelOne.json
```

JSON request body: Request body is JSON of following Go structure

```
{
  ModelName   string           // model name to run
  ModelDigest string           // model digest to run
  RunStamp    string           // run stamp, if empty then auto-generated as timestamp
  Dir         string           // working directory to run the model, if relative then must be relative to oms root directory
  Opts        map[string]string // model run options, e.g.: -OpenM.SubValues 16
  Env         map[string]string // environment variables to set
  Threads     int              // number of modelling threads
  IsMpi       bool             // if true then it use MPI to run the model
  Mpi         struct {
    Np          int    // if non-zero then number of MPI processes
    IsNotOnRoot bool   // if true then do no run modelling threads on MPI root process
    IsNotByJob  bool   // if true then do not allocate resources by job, use CPU, threads and memory as is
  }
  Template  string     // template file name to make run model command line
  Tables    []string   // if not empty then output tables or table groups to retain, by default retain all tables
  Microdata struct {
    IsToDb     bool       // if true then store entity microdata in database: -Microdata.ToDb true
    IsInternal bool       // if true then allow to use internal attributes: -Microdata.UseInternal true
    Entity     []struct { // list of entities and attributes: -Microdata.Person age,income -Microdata.Other All
      Name  string        // entity name
      Attr []string       // list of microdata attributes, it is also can be All
    }
  }
  RunNotes []struct {
    LangCode string   // model language code
    Note     string   // run notes
  }
}
```

Template is a name of template file inside of `etc` sub-directory to make model run command line. Template file is required only if you want to run the model on MPI cluster, when `Mpi.Np > 0`. If template file name not specified then by default it is: `etc/mpiModelRun.template.txt`.

JSON response example:

```
{
  "ModelName":      "modelOne",
  "ModelDigest":    "_201208171604590148_",
  "RunStamp":       "2019_01_29_21_02_14_452",
  "SubmitStamp":    "2019_01_29_21_02_14_448",
  "IsFinal":        false,
  "UpdateDateTime": "2019-01-29 21:02:14.452",
  "RunName":        "",
  "TaskRunName":    ""
}
```

IsFinal: if true then model run failed to start.

RunStamp: model run stamp, use it to GET model run status and log or to PUT stop model run.

Model console output redirected to log file: `models/log/modelName.RunStamp.console.log`, for example: `modelOne.2019_01_29_21_02_14_452.console.log`.

Example 1:

Run modelOne.exe with 2 sub-values (sub-value is similar to Modgen "sub-sample"):

```
{
  "ModelName": "modelOne",
  "Opts": {
    "OpenM.SubValues": "2"
  }
}
```

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/run -d @run_modelOne.json

*   Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 4040 (#0)
> POST /api/run HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.54.1
> Accept: */*
> Content-Type: application/json
> Content-Length: 68
>
* upload completely sent off: 68 out of 68 bytes
< HTTP/1.1 200 OK
< Content-Location: /api/model/_201208171604590148_/run/2019_01_29_21_02_14_452
< Content-Type: application/json
< Date: Wed, 30 Jan 2019 02:02:14 GMT
< Content-Length: 188
<
{
  "ModelName":      "modelOne",
  "ModelDigest":    "_201208171604590148_",
  "RunStamp":       "2019_01_29_21_02_14_452",
  "SubmitStamp":    "2019_01_29_21_02_14_448",
  "IsFinal":        false,
  "UpdateDateTime": "2019-01-29 21:02:14.452",
  "RunName":        "",
  "TaskRunName":    ""
}
* Connection #0 to host localhost left intact
```

Oms web-service execute following command:

```
./modelOne -OpenM.RunStamp 2019_01_29_21_02_14_452 -OpenM.LogToConsole true -OpenM.SubValues 2
```

As result `modelOne` executable started on server with 2 sub-values. Model console output redirected to log file `modelOne.2019_01_29_21_02_14_452.console.log`:

```
2019-01-29 21:02:14.469 modelOne
2019-01-29 21:02:14.486 Run: 138
2019-01-29 21:02:14.486 Reading Parameters
2019-01-29 21:02:14.487 Running Simulation
2019-01-29 21:02:14.487 Writing Output Tables
2019-01-29 21:02:14.567 Running Simulation
2019-01-29 21:02:14.567 Writing Output Tables
2019-01-29 21:02:14.622 Done.
```

Example 2:

Run RiskPaths model in `models/work` directory:

```
{
  "ModelName": "RiskPaths",
  "Dir": "models/work",
  "Opts": {
    "OpenM.Database": "Database=../bin/RiskPaths.sqlite;OpenMode=ReadWrite;Timeout=86400;"
  },
  "RunNotes": [{
      "LangCode": "EN",
      "Note": "Model run notes.\n----------------\n\nThis is model run notes in English"
    }, {
      "LangCode": "FR",
      "Note": "(FR) Model run notes.\n---------------------\n\nJe suis désolé je ne parle pas français"
    }
  ]
}
```

Oms web-service execute following commands:

```
cd models/work
../bin/RiskPaths \
  -OpenM.RunStamp 2019_01_29_21_32_41_179 \
  -OpenM.LogToConsole true \
  -OpenM.Database Database=../bin/RiskPaths.sqlite;OpenMode=ReadWrite;Timeout=86400; \
  -EN.2019_01_29_21_32_41_179.run_notes.EN.md \
  -FR.2019_01_29_21_32_41_179.run_notes.EN.md
```

Example 3:

Run RiskPaths_mpi model executable on two nodes of small MPI cluster, 4 threads on each node, to calculate 16 sub-values:

```
{
  "ModelName": "RiskPaths",
  "Opts": {
    "OpenM.SubValues": "16"
  },
  "Threads": 4,
  "Mpi": {
    "Np": 2
  },
  "Template": "mpiSmallCluster.template.txt"
}
```

Oms web-service execute following commands:

```
mpirun -n 2 -wdir models/bin ./RiskPaths_mpi -OpenM.RunStamp 2019_01_29_21_32_10_577 -OpenM.LogToConsole true -OpenM.Threads 4 -OpenM.SubValues 16
```

Because `Mpi.Np = 2` model is executed on MPI cluster. If we do not specify template file name `mpiSmallCluster.template.txt` then by default `etc/mpiModelRun.template.txt` will be used.

Example 4:

Run OzProj model, which may required `OM_OzProj` environment variable:

```
{
  "ModelName": "OzProj",
  "RunStamp": "My-uniqueStamp-of-OzProj-run",
  "Env": {
    "OM_OzProj": "../../OzProj"
  },
  "Opts": {
    "OpenM.ProgressPercent": "25"
  }
}
```

Oms web-service execute following commands:

```
OM_OzProj=../../OzProj ./OzProj -OpenM.RunStamp My-uniqueStamp-of-OzProj-run -OpenM.LogToConsole true -OpenM.ProgressPercent 25
```

Because `RunStamp` explicitly specified model console output log file name is: `OzProj.My-uniqueStamp-of-OzProj-run.console.log`. It is strongly recommended to use unique run stamps for each model run (modeling task run, if you running modeling task).

GET state of current model run

Get status of model run and view model console output.

This method allow get current model run and model stdout and stderr. It can be used to monitor modeling progress or it can be invoke later to see final model run status and console output.

This is a beta version and may change in the future.

Method:

```
GET /api/run/log/model/:model/stamp/:stamp
GET /api/run/log/model/:model/stamp/:stamp/start/:start/count/:count
```

Call examples:

```
http://localhost:4040/api/run/log/model/modelOne/stamp/2016_08_17_21_07_55_123
http://localhost:4040/api/run/log/model/modelOne/stamp/My-own-run-uniqueStamp
http://localhost:4040/api/run/log/model/modelOne/stamp/My-own-run-uniqueStamp/start/0
http://localhost:4040/api/run/log/model/modelOne/stamp/My-own-run-uniqueStamp/start/0/count/100
```

Arguments as URL parameters:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:stamp - (required) model run stamp or run submission stamp or modeling task run stamp
```

Model run identified by submission stamp or by run stamp. Submission stamp is automatically generated by server as timestamp string, e.g.: 2016_08_17_21_07_55_111. Run stamp can be either explicitly specified as part of request to run the model call or automatically generated as timestamp string. By default oms service store in memory history of 100 most recent model runs (it can be configured).

```
:start - (optional) start "page" line number from log output, zero-based.
:count - (optional) "page" size, number of log text lines to view.
```

By default oms service selects 100 lines (it can be configured). If count <= 0 specified then all lines selected.

Example:


```
{
  "ModelName": "modelOne",
  "ModelDigest": "_201208171604590148_",
  "RunStamp": "2019_01_29_20_03_58_681",
  "SubmitStamp": "2019_01_29_20_03_58_677",
  "IsFinal": true,
  "UpdateDateTime": "2019-01-29 20:03:58.818",
  "RunName": "",
  "TaskRunName": "",
  "Offset": 0,
  "Size": 6,
  "TotalSize": 6,
  "Lines": [
    "2019-01-29 20:03:58.694 modelOne",
    "2019-01-29 20:03:58.712 Run: 135",
    "2019-01-29 20:03:58.712 Reading Parameters",
    "2019-01-29 20:03:58.713 Running Simulation",
    "2019-01-29 20:03:58.713 Writing Output Tables",
    "2019-01-29 20:03:58.809 Done."
  ]
}
```


IsFinal: if true then model run completed.

PUT stop model run

Stop model run by killing the process.

This method allow to stop model run by sending kill signal to the model process (or to the leading process in case of MPI model run).

This is a beta version and may change in the future.

Method:

```
PUT /api/run/stop/model/:model/stamp/:stamp
```

Call examples:

```
http://localhost:4040/api/run/stop/model/modelOne/stamp/2016_08_17_21_07_55_123
http://localhost:4040/api/run/stop/model/modelOne/stamp/My-own-run-uniqueStamp
```

Arguments as URL parameters:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:stamp - (required) model run stamp or modeling task run stamp
```

Model run identified by run stamp, which either explicitly specified as part of request to run the model call or automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123.

Example:

```
curl -v -X PUT http://localhost:4040/api/run/stop/model/dd41bca43ea03484916be3088957f2ce/stamp/2022_06_07_16_25_30_105

*   Trying 127.0.0.1:4040...
* Connected to localhost (127.0.0.1) port 4040 (#0)
> PUT /api/run/stop/model/dd41bca43ea03484916be3088957f2ce/stamp/2022_06_07_16_25_30_105 HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.79.1
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Content-Location: /api/model/dd41bca43ea03484916be3088957f2ce/run/2022_06_07_16_25_30_105/true
< Date: Tue, 07 Jun 2022 20:25:52 GMT
< Content-Length: 0
<
* Connection #0 to host localhost left intact
```

GET download log file

GET download log file from download directory on the server.

This method only avaliable if server configured to create downloads for user.

This is a beta version and may change in the future.

Download can be initiated through UI or by direct API call:

- POST initiate model download
- POST initiate model run download
- POST initiate workset download

For each download `oms` service does create a download log file, for example:

- `RiskPaths.run.102.ready.download.log` RiskPaths model run results, download ready for user
- `RiskPaths.run.102.progress.download.log` RiskPaths model run results, download preparation in progress
- `RiskPaths.run.102.error.download.log` RiskPaths model run results, download preparation failed

Example of `RiskPaths.run.102.ready.download.log`:

```
2021-07-31 18:13:10.293 Download of: RiskPaths.run.102
---------------
Model Name    : RiskPaths
Model Version : 3.0.0.0 2021-07-16 13:14:14.451
Model Digest  : 0f71660ba32bc002282c995e4552a14a
Run Name      : Default
Run Version   : 102 2021-07-16 13:14:22.227
Run Digest    : 4354632979ec90f48441ccdeb0ca803b
Folder        : RiskPaths.run.102
---------------
2021-07-31 18:13:10.293 delete: RiskPaths.run.102.ready.download.log
2021-07-31 18:13:10.293 delete: RiskPaths.run.102.error.download.log
2021-07-31 18:13:10.293 delete: RiskPaths.run.102.zip
2021-07-31 18:13:10.293 delete: RiskPaths.run.102
2021-07-31 18:13:10.330 Model RiskPaths
2021-07-31 18:13:10.339 Model run 102 Default
2021-07-31 18:13:10.401 Packed C:\go_ws\models\home\out\download\RiskPaths.run.102.zip
2021-07-31 18:13:10.402 Done.
```

As result `oms` service does create:

- download archive, for example: `RiskPaths.run.102.zip`
- model run `.csv` files for parameters and output tables in `RiskPaths.run.102` folder
- model run `.json` metadata files

Method:

```
GET /api/download/log/file/:name
```

Arguments:

```
:name - (required) download log file name, for example: `RiskPaths.run.102.ready.download.log`
```

Call example from browser:

```
http://localhost:4040/api/download/log/file/RiskPaths.run.102.ready.download.log
```

Return example:

```
{
  "Status": "ready",
  "Kind": "run",
  "ModelDigest": "0f71660ba32bc002282c995e4552a14a",
  "RunDigest": "4354632979ec90f48441ccdeb0ca803b",
  "WorksetName": "",
  "IsFolder": true,
  "Folder": "RiskPaths.run.102",
  "IsZip": true,
  "ZipFileName": "RiskPaths.run.102.zip",
  "ZipModTime": 1627769590401,
  "ZipSize": 16525,
  "LogFileName": "RiskPaths.run.102.ready.download.log",
  "LogNsTime": 1627769590402950000,
  "Lines": [
    "2021-07-31 18:13:10.293 Download of: RiskPaths.run.102 ",
    "--------------- ",
    "Model Name    : RiskPaths ",
    "Model Version : 3.0.0.0 2021-07-16 13:14:14.451 ",
    "Model Digest  : 0f71660ba32bc002282c995e4552a14a ",
    "Run Name      : Default ",
    "Run Version   : 102 2021-07-16 13:14:22.227 ",
    "Run Digest    : 4354632979ec90f48441ccdeb0ca803b ",
    "Folder        : RiskPaths.run.102 ",
    "--------------- ",
    "2021-07-31 18:13:10.293 delete: RiskPaths.run.102.ready.download.log ",
    "2021-07-31 18:13:10.293 delete: RiskPaths.run.102.error.download.log ",
    "2021-07-31 18:13:10.293 delete: RiskPaths.run.102.zip ",
    "2021-07-31 18:13:10.293 delete: RiskPaths.run.102 ",
    "2021-07-31 18:13:10.330 Model RiskPaths  ",
    "2021-07-31 18:13:10.339 Model run 102 Default ",
    "2021-07-31 18:13:10.401 Packed C:\\go_ws\\models\\home\\out\\download\\RiskPaths.run.102.zip ",
    "2021-07-31 18:13:10.402 Done. ",
    ""
  ]
}
```

GET model download log files

GET all model downloads log files from download directory on the server.

This method only avaliable if server configured to create downloads for user.

This is a beta version and may change in the future.

Download can be initiated through UI or by direct API call:

- POST initiate model download
- POST initiate model run download
- POST initiate model download

For each download `oms` service does create a download log file, for example:

- `RiskPaths.run.102.ready.download.log` RiskPaths model run results, download ready for user
- `RiskPaths.run.102.progress.download.log` RiskPaths model run results, download preparation in progress
- `RiskPaths.run.102.error.download.log` RiskPaths model run results, download preparation failed

Example of `RiskPaths.run.102.ready.download.log`:

```
2021-07-31 18:13:10.293 Download of: RiskPaths.run.102
---------------
Model Name    : RiskPaths
Model Version : 3.0.0.0 2021-07-16 13:14:14.451
Model Digest  : 0f71660ba32bc002282c995e4552a14a
Run Name      : Default
Run Version   : 102 2021-07-16 13:14:22.227
Run Digest    : 4354632979ec90f48441ccdeb0ca803b
Folder        : RiskPaths.run.102
---------------
2021-07-31 18:13:10.293 delete: RiskPaths.run.102.ready.download.log
2021-07-31 18:13:10.293 delete: RiskPaths.run.102.error.download.log
2021-07-31 18:13:10.293 delete: RiskPaths.run.102.zip
2021-07-31 18:13:10.293 delete: RiskPaths.run.102
2021-07-31 18:13:10.330 Model RiskPaths
2021-07-31 18:13:10.339 Model run 102 Default
2021-07-31 18:13:10.401 Packed C:\go_ws\models\home\out\download\RiskPaths.run.102.zip
2021-07-31 18:13:10.402 Done.
```

As result `oms` service does create:

- download archive, for example: `RiskPaths.run.102.zip`
- model run `.csv` files for parameters and output tables in `RiskPaths.run.102` folder
- model run `.json` metadata files

Method:

```
GET /api/download/log/model/:model
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database then result is undefined.

Call examples from browser:

```
http://localhost:4040/api/download/log/model/RiskPaths
http://localhost:4040/api/download/log/model/0f71660ba32bc002282c995e4552a14a
```

Return example:

```
[
  {
    "Status": "ready",
    "Kind": "model",
    "ModelDigest": "0f71660ba32bc002282c995e4552a14a",
    "RunDigest": "",
    "WorksetName": "",
    "IsFolder": true,
    "Folder": "RiskPaths",
    "IsZip": true,
    "ZipFileName": "RiskPaths.zip",
    "ZipModTime": 1628178307162,
    "ZipSize": 30323,
    "LogFileName": "RiskPaths.ready.download.log",
    "LogNsTime": 0,
    "Lines": [
      "2021-08-05 11:45:06.371 Download of: RiskPaths ",
      "--------------- ",
      "Model Name    : RiskPaths ",
      "Model Version : 3.0.0.0 2021-08-02 14:16:34.584 ",
      "Model Digest  : 0f71660ba32bc002282c995e4552a14a ",
      "Folder        : RiskPaths ",
      "--------------- ",
      "2021-08-05 11:45:06.371 delete: RiskPaths.ready.download.log ",
      "2021-08-05 11:45:06.373 delete: RiskPaths.error.download.log ",
      "2021-08-05 11:45:06.373 delete: RiskPaths.zip ",
      "2021-08-05 11:45:06.378 delete: RiskPaths ",
      "2021-08-05 11:45:07.025 Model RiskPaths  ",
      "2021-08-05 11:45:07.056 Model run 102 RiskPaths_Default ",
      "2021-08-05 11:45:07.129 Workset 101 Default ",
      "2021-08-05 11:45:07.162 Packed C:\\go_ws\\models\\home\\out\\download\\RiskPaths.zip ",
      "2021-08-05 11:45:07.163 Done. ",
      ""
    ]
  },
  {
    "Status": "ready",
    "Kind": "run",
    "ModelDigest": "0f71660ba32bc002282c995e4552a14a",
    "RunDigest": "4354632979ec90f48441ccdeb0ca803b",
    "WorksetName": "",
    "IsFolder": true,
    "Folder": "RiskPaths.run.102",
    "IsZip": true,
    "ZipFileName": "RiskPaths.run.102.zip",
    "ZipModTime": 1627769590401,
    "ZipSize": 16525,
    "LogFileName": "RiskPaths.run.102.ready.download.log",
    "LogNsTime": 0,
    "Lines": [
      "2021-07-31 18:13:10.293 Download of: RiskPaths.run.102 ",
      "--------------- ",
      "Model Name    : RiskPaths ",
      "Model Version : 3.0.0.0 2021-07-16 13:14:14.451 ",
      "Model Digest  : 0f71660ba32bc002282c995e4552a14a ",
      "Run Name      : Default ",
      "Run Version   : 102 2021-07-16 13:14:22.227 ",
      "Run Digest    : 4354632979ec90f48441ccdeb0ca803b ",
      "Folder        : RiskPaths.run.102 ",
      "--------------- ",
      "2021-07-31 18:13:10.293 delete: RiskPaths.run.102.ready.download.log ",
      "2021-07-31 18:13:10.293 delete: RiskPaths.run.102.error.download.log ",
      "2021-07-31 18:13:10.293 delete: RiskPaths.run.102.zip ",
      "2021-07-31 18:13:10.293 delete: RiskPaths.run.102 ",
      "2021-07-31 18:13:10.330 Model RiskPaths  ",
      "2021-07-31 18:13:10.339 Model run 102 Default ",
      "2021-07-31 18:13:10.401 Packed C:\\go_ws\\models\\home\\out\\download\\RiskPaths.run.102.zip ",
      "2021-07-31 18:13:10.402 Done. ",
      ""
    ]
  },
  {
    "Status": "ready",
    "Kind": "workset",
    "ModelDigest": "0f71660ba32bc002282c995e4552a14a",
    "RunDigest": "",
    "WorksetName": "Default",
    "IsFolder": true,
    "Folder": "RiskPaths.set.Default",
    "IsZip": true,
    "ZipFileName": "RiskPaths.set.Default.zip",
    "ZipModTime": 1627770244509,
    "ZipSize": 3691,
    "LogFileName": "RiskPaths.set.Default.ready.download.log",
    "LogNsTime": 0,
    "Lines": [
```

```
    "Lines": [
      "2021-07-31 18:24:04.069 Download of: RiskPaths.set.Default ",
      "------------------ ",
      "Model Name       : RiskPaths ",
      "Model Version    : 3.0.0.0 2021-07-16 13:14:14.451 ",
      "Model Digest     : 0f71660ba32bc002282c995e4552a14a ",
      "Scenario Name    : Default ",
      "Scenario Version : 2021-07-30 01:58:34.496 ",
      "Folder           : RiskPaths.set.Default ",
      "------------------ ",
      "2021-07-31 18:24:04.071 delete: RiskPaths.set.Default.ready.download.log ",
      "2021-07-31 18:24:04.071 delete: RiskPaths.set.Default.error.download.log ",
      "2021-07-31 18:24:04.071 delete: RiskPaths.set.Default.zip ",
      "2021-07-31 18:24:04.074 delete: RiskPaths.set.Default ",
      "2021-07-31 18:24:04.461 Model RiskPaths  ",
      "2021-07-31 18:24:04.469 Workset 101 Default ",
      "2021-07-31 18:24:04.509 Packed C:\\go_ws\\models\\home\\out\\download\\RiskPaths.set.Default.zip ",
      "2021-07-31 18:24:04.510 Done. ",
      ""
    ]
  }
]
```

GET all download log files

GET all downloads log files for from download directory on the server.

This method only avaliable if server configured to create downloads for user.

This is a beta version and may change in the future.

Download can be initiated through UI or by direct API call:

- POST initiate model download
- POST initiate model run download
- POST initiate model download

For each download `oms` service does create a download log file, for example:

- `RiskPaths.run.102.ready.download.log` RiskPaths model run results, download ready for user
- `RiskPaths.run.102.progress.download.log` RiskPaths model run results, download preparation in progress
- `RiskPaths.run.102.error.download.log` RiskPaths model run results, download preparation failed

Example of `RiskPaths.run.102.ready.download.log`:

```
2021-07-31 18:13:10.293 Download of: RiskPaths.run.102
---------------
Model Name    : RiskPaths
Model Version : 3.0.0.0 2021-07-16 13:14:14.451
Model Digest  : 0f71660ba32bc002282c995e4552a14a
Run Name      : Default
Run Version   : 102 2021-07-16 13:14:22.227
Run Digest    : 4354632979ec90f48441ccdeb0ca803b
Folder        : RiskPaths.run.102
---------------
2021-07-31 18:13:10.293 delete: RiskPaths.run.102.ready.download.log
2021-07-31 18:13:10.293 delete: RiskPaths.run.102.error.download.log
2021-07-31 18:13:10.293 delete: RiskPaths.run.102.zip
2021-07-31 18:13:10.293 delete: RiskPaths.run.102
2021-07-31 18:13:10.330 Model RiskPaths
2021-07-31 18:13:10.339 Model run 102 Default
2021-07-31 18:13:10.401 Packed C:\go_ws\models\home\out\download\RiskPaths.run.102.zip
2021-07-31 18:13:10.402 Done.
```

As result `oms` service does create:

- download archive, for example: `RiskPaths.run.102.zip`
- model run `.csv` files for parameters and output tables in `RiskPaths.run.102` folder
- model run `.json` metadata files

Method:

```
GET /api/download/log/all
```

Call example from browser:

```
http://localhost:4040/api/download/log/all
```

Return example:

```
[
  {
    "Status": "ready",
    "Kind": "model",
    "ModelDigest": "c87bd08cc86da61332336384a491203b",
    "RunDigest": "",
    "WorksetName": "",
    "IsFolder": true,
    "Folder": "IDMM",
    "IsZip": true,
    "ZipFileName": "IDMM.zip",
    "ZipModTime": 1627790748053,
    "ZipSize": 29126,
    "LogFileName": "IDMM.ready.download.log",
    "LogNsTime": 0,
    "Lines": [
      "2021-08-01 00:05:47.551 Download of: IDMM ",
      "--------------- ",
      "Model Name    : IDMM ",
      "Model Version : 2.0.0.0 2021-07-16 13:13:40.085 ",
      "Model Digest  : c87bd08cc86da61332336384a491203b ",
      "Folder        : IDMM ",
      "--------------- ",
      "2021-08-01 00:05:47.551 delete: IDMM.ready.download.log ",
      "2021-08-01 00:05:47.552 delete: IDMM.error.download.log ",
      "2021-08-01 00:05:47.553 delete: IDMM.zip ",
      "2021-08-01 00:05:47.553 delete: IDMM ",
      "2021-08-01 00:05:47.934 Model IDMM  ",
      "2021-08-01 00:05:47.946 Model run 102 Default ",
      "2021-08-01 00:05:47.968 Model run 103 IDMM_Default_2021_07_31_21_40_28_624 ",
      "2021-08-01 00:05:47.990 Workset 101 Default ",
      "2021-08-01 00:05:48.053 Packed C:\\go_ws\\models\\home\\out\\download\\IDMM.zip ",
      "2021-08-01 00:05:48.054 Done. ",
      ""
    ]
  },
  {
    "Status": "ready",
    "Kind": "model",
    "ModelDigest": "b4f2100f8d308a5bd3bf3b470077d906",
    "RunDigest": "",
    "WorksetName": "",
    "IsFolder": true,
    "Folder": "NewTimeBased",
    "IsZip": true,
    "ZipFileName": "NewTimeBased.zip",
    "ZipModTime": 1627848086688,
    "ZipSize": 8199,
    "LogFileName": "NewTimeBased.ready.download.log",
    "LogNsTime": 0,
    "Lines": [
      "2021-08-01 16:01:26.574 Download of: NewTimeBased ",
      "--------------- ",
      "Model Name    : NewTimeBased ",
      "Model Version : 1.0.1.0 2021-07-16 13:14:32.196 ",
      "Model Digest  : b4f2100f8d308a5bd3bf3b470077d906 ",
      "Folder        : NewTimeBased ",
      "--------------- ",
      "2021-08-01 16:01:26.574 delete: NewTimeBased.ready.download.log ",
      "2021-08-01 16:01:26.574 delete: NewTimeBased.error.download.log ",
      "2021-08-01 16:01:26.574 delete: NewTimeBased.zip ",
      "2021-08-01 16:01:26.574 delete: NewTimeBased ",
      "2021-08-01 16:01:26.610 Model NewTimeBased  ",
      "2021-08-01 16:01:26.641 Model run 102 Default ",
      "2021-08-01 16:01:26.666 Workset 101 Default ",
      "2021-08-01 16:01:26.689 Packed C:\\go_ws\\models\\home\\out\\download\\NewTimeBased.zip ",
      "2021-08-01 16:01:26.693 Done. ",
      ""
    ]
  }
]
```

GET download files tree

GET download files tree from download directory on the server.

This method only avaliable if server configured to create downloads for user.

This is a beta version and may change in the future.

Download can be initiated through UI or by direct API call:

- POST initiate model download
- POST initiate model run download
- POST initiate model download

For each above method `oms` service will create:

- download archive, for example: `RiskPaths.run.102.zip`
- model run `.csv` files for parameters and output tables in `RiskPaths.run.102` folder
- model run `.json` metadata files

This method retruns file tree in download folder, for exmaple in `RiskPaths.run.102` folder.

Method:

```
GET /api/download/file-tree/:folder
```

Arguments:

```
:folder - (required) download folder file name, for example: `RiskPaths.run.102`
```

Call example from browser:

```
http://localhost:4040/api/download/file-tree/RiskPaths.run.102
```

Return example:


```
[
 {
  "Path": "RiskPaths.run.102",
  "IsDir": true,
  "Size": 0,
  "ModTime": 1627769590376
 },
 {
  "Path": "RiskPaths.run.102/RiskPaths.run.102.Default.json",
  "IsDir": false,
  "Size": 1880,
  "ModTime": 1627769590376
 },
 {
  "Path": "RiskPaths.run.102/run.102.Default",
  "IsDir": true,
  "Size": 0,
  "ModTime": 1627769590374
 },
 {
  "Path": "RiskPaths.run.102/run.102.Default/AgeBaselineForm1.csv",
  "IsDir": false,
  "Size": 283,
  "ModTime": 1627769590340
 },
 {
  "Path": "RiskPaths.run.102/run.102.Default/AgeBaselinePreg1.csv",
  "IsDir": false,
  "Size": 265,
  "ModTime": 1627769590341
 },
 {
  "Path": "RiskPaths.run.102/run.102.Default/CanDie.csv",
```

"IsDir": false,
"Size": 27,
"ModTime": 1627769590344
},
{
"Path": "RiskPaths.run.102/run.102.Default/ProbMort.csv",
"IsDir": false,
"Size": 1022,
"ModTime": 1627769590347
},
{
"Path": "RiskPaths.run.102/run.102.Default/SeparationDurationBaseline.csv",
"IsDir": false,
"Size": 133,
"ModTime": 1627769590347
},
{
"Path": "RiskPaths.run.102/run.102.Default/SimulationCases.csv",
"IsDir": false,
"Size": 26,
"ModTime": 1627769590349
},
{
"Path": "RiskPaths.run.102/run.102.Default/SimulationSeed.csv",
"IsDir": false,
"Size": 23,
"ModTime": 1627769590350
},
{
"Path": "RiskPaths.run.102/run.102.Default/T01_LifeExpectancy.acc-all.csv",
"IsDir": false,
"Size": 65,
"ModTime": 1627769590356
},
{
"Path": "RiskPaths.run.102/run.102.Default/T01_LifeExpectancy.acc.csv",
"IsDir": false,
"Size": 52,
"ModTime": 1627769590352
},
{
"Path": "RiskPaths.run.102/run.102.Default/T01_LifeExpectancy.csv",
"IsDir": false,
"Size": 55,
"ModTime": 1627769590352
},
{
"Path": "RiskPaths.run.102/run.102.Default/T02_TotalPopulationByYear.acc-all.csv",
"IsDir": false,
"Size": 2544,
"ModTime": 1627769590361
},
{
"Path": "RiskPaths.run.102/run.102.Default/T02_TotalPopulationByYear.acc.csv",
"IsDir": false,
"Size": 3040,
"ModTime": 1627769590359
},
{
"Path": "RiskPaths.run.102/run.102.Default/T02_TotalPopulationByYear.csv",
"IsDir": false,
"Size": 2833,
"ModTime": 1627769590357
},
{
"Path": "RiskPaths.run.102/run.102.Default/T03_FertilityByAge.acc-all.csv",
"IsDir": false,
"Size": 1665,
"ModTime": 1627769590364
},
{
"Path": "RiskPaths.run.102/run.102.Default/T03_FertilityByAge.acc.csv",
"IsDir": false,
"Size": 2080,
"ModTime": 1627769590362
},
{
"Path": "RiskPaths.run.102/run.102.Default/T03_FertilityByAge.csv",
"IsDir": false,
"Size": 1532,
"ModTime": 1627769590362
},
{
"Path": "RiskPaths.run.102/run.102.Default/T04_FertilityRatesByAgeGroup.acc-all.csv",
"IsDir": false,
"Size": 4583,
"ModTime": 1627769590367
},

```
  {
    "Path": "RiskPaths.run.102/run.102.Default/T04_FertilityRatesByAgeGroup.acc.csv",
    "IsDir": false,
    "Size": 6905,
    "ModTime": 1627769590366
  },
  {
    "Path": "RiskPaths.run.102/run.102.Default/T04_FertilityRatesByAgeGroup.csv",
    "IsDir": false,
    "Size": 3676,
    "ModTime": 1627769590365
  },
  {
    "Path": "RiskPaths.run.102/run.102.Default/T05_CohortFertility.acc-all.csv",
    "IsDir": false,
    "Size": 100,
    "ModTime": 1627769590369
  },
  {
    "Path": "RiskPaths.run.102/run.102.Default/T05_CohortFertility.acc.csv",
    "IsDir": false,
    "Size": 74,
    "ModTime": 1627769590368
  },
  {
    "Path": "RiskPaths.run.102/run.102.Default/T05_CohortFertility.csv",
    "IsDir": false,
    "Size": 70,
    "ModTime": 1627769590368
  },
  {
    "Path": "RiskPaths.run.102/run.102.Default/T06_BirthsByUnion.acc-all.csv",
    "IsDir": false,
    "Size": 218,
    "ModTime": 1627769590371
  },
  {
    "Path": "RiskPaths.run.102/run.102.Default/T06_BirthsByUnion.acc.csv",
    "IsDir": false,
    "Size": 234,
    "ModTime": 1627769590371
  },
  {
    "Path": "RiskPaths.run.102/run.102.Default/T06_BirthsByUnion.csv",
    "IsDir": false,
    "Size": 222,
    "ModTime": 1627769590370
  },
  {
    "Path": "RiskPaths.run.102/run.102.Default/T07_FirstUnionFormation.acc-all.csv",
    "IsDir": false,
    "Size": 604,
    "ModTime": 1627769590374
  },
  {
    "Path": "RiskPaths.run.102/run.102.Default/T07_FirstUnionFormation.acc.csv",
    "IsDir": false,
    "Size": 707,
    "ModTime": 1627769590374
  },
  {
    "Path": "RiskPaths.run.102/run.102.Default/T07_FirstUnionFormation.csv",
    "IsDir": false,
    "Size": 428,
    "ModTime": 1627769590373
  },
  {
    "Path": "RiskPaths.run.102/run.102.Default/UnionDurationBaseline.csv",
    "IsDir": false,
    "Size": 395,
    "ModTime": 1627769590350
  },
  {
    "Path": "RiskPaths.run.102/run.102.Default/UnionStatusPreg1.csv",
    "IsDir": false,
    "Size": 196,
    "ModTime": 1627769590352
  }
]
```

POST initiate entire model download

POST model download request: server will prepare entire model data for download.

This method only avaliable if server configured to create downloads for user.

This is a beta version and may change in the future.

As result of this call `oms` service will create:

- download archive, for example: `RiskPaths.zip`
- model run `.csv` files for parameters and output tables in `RiskPaths` folder
- model run `.json` metadata files

Method:

```
POST /api/download/model/:model
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

Form body

Posted form can include optional JSON body:

```
{
  "NoAccumulatorsCsv": true,
  "NoMicrodata": true,
  "Utf8BomIntoCsv": true
}
```

Use `"NoAccumulatorsCsv": true` and `"NoMicrodata": true` options to produce download faster. By default this method create full Model.zip archive, which allow you to copy model into desktop database by using dbcopy utilty or even transfer it to the other server.

If you want only to analyze model run output CSV files then it maybe better to download run results without accumulators (a.k.a. sub-samples or sub-values) and include only output table expressions. For example, if you are only interested in output average value and don't want to analyze 32 sub-samples then use `"NoAccumulatorsCsv": true` option.

Also model run microdata can be huge and if you are not intersted in it then use `"NoMicrodata": true` option to suppress it.

Use `"Utf8BomIntoCsv": true` option to start CSV files with Byte Order Mark. Byte order mark may be neccessary for some programs (e.g. Microsoft Excel) to correctly process UTF-8 files.

Call examples:

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/download/model/modelOne
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/download/model/c87bd08cc86da61332336384a491203b

curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/download/model/modelOne -d @options.json
```

POST initiate model run download

POST model run download request: server will prepare model run data for download.

This method only avaliable if server configured to create downloads for user.

This is a beta version and may change in the future.

As result of this call `oms` service will create:

- download archive, for example: `RiskPaths.run.102.zip`
- model run `.csv` files for parameters and output tables in `RiskPaths.run.102` folder
- model run `.json` metadata files

Method:

```
POST /api/download/model/:model/run/:run
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (required) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run. Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined.

Form body

Posted form can include optional JSON body:

```
{
  "NoAccumulatorsCsv": true,
  "NoMicrodata": true,
  "Utf8BomIntoCsv": true
}
```

Use `"NoAccumulatorsCsv": true` and `"NoMicrodata": true` options to produce download faster. By default this method create full ModelRun.zip archive, which allow you to copy model into desktop database by using dbcopy utilty or even transfer it to the other server.

If you want only to analyze model run output CSV files then it maybe better to download run results without accumulators (a.k.a. sub-samples or sub-values) and include only output table expressions. For example, if you are only interested in output average value and don't want to analyze 32 sub-samples then use `"NoAccumulatorsCsv": true` option.

Also model run microdata can be huge and if you are not intersted in it then use `"NoMicrodata": true` option to suppress it.

Use `"Utf8BomIntoCsv": true` option to start CSV files with Byte Order Mark. Byte order mark may be neccessary for some programs (e.g. Microsoft Excel) to correctly process UTF-8 files.

Call examples:

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/download/model/modelOne/run/Default
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/download/model/c87bd08cc86da61332336384a491203b/run/Default
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/download/model/c87bd08cc86da61332336384a491203b/run/D3f26c4492bad08b9d6c8373719ff53e7

curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/download/model/modelOne/run/Default -d @options.json
```

POST initiate model workset download

POST model workset download request: server will prepare model workset data for download.

This method only avaliable if server configured to create downloads for user.

This is a beta version and may change in the future.

Workset is a set of model input parameters (a.k.a. "scenario" input). Workset can be used to run the model.

As result of this call `oms` service will create:

- download archive, for example: `RiskPaths.set.Default.zip`
- model workset `.csv` files for parameters and output tables in `RiskPaths.set.Default` folder
- model workset `.json` metadata files

Method:

```
POST /api/download/model/:model/workset/:set
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:set - (required) workset name
```

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default".

Form body

Posted form can include optional JSON body:

```
{
  "Utf8BomIntoCsv": true
}
```

Use `"Utf8BomIntoCsv": true` option to start CSV files with Byte Order Mark. Byte order mark may be neccessary for some programs (e.g. Microsoft Excel) to correctly process UTF-8 files.

Call examples:

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/download/model/RiskPaths/workset/Default
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/download/model/c87bd08cc86da61332336384a491203b/workset/Default

curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/download/model/RiskPaths/workset/Default -d @options.json
```

DELETE download files

DELETE download files from the server.

This method only avaliable if server configured to create downloads for user.

This is a beta version and may change in the future.

Download can be initiated through UI or by direct API call:

- POST initiate model download
- POST initiate model run download
- POST initiate model download

For each above method `oms` service will create:

- download archive, for example: `RiskPaths.run.102.zip`
- model run `.csv` files for parameters and output tables in `RiskPaths.run.102` folder
- model run `.json` metadata files

This method deletes download files from the server.

Method:

```
DELETE /api/download/delete/:folder
```

Arguments:

```
:folder - (required) download folder file name, for example: `RiskPaths.run.102`
```

Call example:

```
curl -v -X DELETE http://localhost:4040/api/download/delete/RiskPaths.run.102
```

DELETE download files asynchronously

DELETE download files from the server asynchronously.

This method only avaliable if server configured to create downloads for user.

This is a beta version and may change in the future.

Download can be initiated through UI or by direct API call:

- POST initiate model download
- POST initiate model run download
- POST initiate model download

For each above method `oms` service will create:

- download archive, for example: `RiskPaths.run.102.zip`
- model run `.csv` files for parameters and output tables in `RiskPaths.run.102` folder
- model run `.json` metadata files

This method will initiate deleting of download files from the server. Actual delete performed in background and results can be checked through log file.

Method:

```
DELETE /api/download/start/delete/:folder
```

Arguments:

```
:folder - (required) download folder file name, for example: `RiskPaths.run.102`
```

Call example:

```
curl -v -X DELETE http://localhost:4040/api/download/start/delete/RiskPaths.run.102
```

GET upload log file

GET upload log file from upload directory on the server.

This method only avaliable if server configured to create uploads for user.

This is a beta version and may change in the future.

Upload can be initiated through UI or by direct API call:

- POST initiate model run upload
- POST initiate workset upload

For each upload `oms` service does create a upload log file, for example:

- `RiskPaths.set.New-Data.ready.upload.log` RiskPaths New-Data workset, upload completed and ready to use
- `RiskPaths.set.New-Data.progress.upload.log` RiskPaths New-Data, upload in progress
- `RiskPaths.set.New-Data.error.upload.log` RiskPaths New-Data, upload failed

Example of `RiskPaths.set.New-Data.ready.upload.log`:

```
2022-03-09 00:21:45.195 Upload of: RiskPaths.set.New-Data
------------------
Upload          : RiskPaths.set.New-Data.zip
Model Name      : RiskPaths
Model Version   : 3.0.0.0 2022-03-07 23:37:41.202
Model Digest    : d90e1e9a49a06d972ecf1d50e684c62b
Scenario Name   : New-Data
Folder          : RiskPaths.set.New-Data
------------------
2022-03-09 00:21:45.195 delete: RiskPaths.set.New-Data.ready.upload.log
2022-03-09 00:21:45.195 delete: RiskPaths.set.New-Data.error.upload.log
2022-03-09 00:21:45.195 delete: RiskPaths.set.New-Data
2022-03-09 00:21:45.195 dbcopy -m RiskPaths -dbcopy.IdOutputNames=false -dbcopy.SetName New-Data -dbcopy.To db -dbcopy.Zip -dbcopy.InputDir models\home\io\upload
2022-03-09 00:21:45.219 Model RiskPaths
2022-03-09 00:21:45.219 Unpack RiskPaths.set.New-Data.zip
2022-03-09 00:21:45.249 Workset New-Data into: 103 New-Data
2022-03-09 00:21:45.249   Parameters: 3
2022-03-09 00:21:45.277 Done.
```

As result of workset upload `oms` service does:

- upload archive, for example: `RiskPaths.set.New-Data.zip`
- extract workset `.csv` files with parameters into `RiskPaths.set.New-Data` folder
- extract workset `.json` metadata file
- create new or update existing New-Data workset in RiskPaths model database

Method:

```
GET /api/upload/log/file/:name
```

Arguments:

```
:name - (required) upload log file name, for example: `RiskPaths.set.New-Data.ready.upload.log`
```

Call example from browser:

```
http://localhost:4040/api/upload/log/file/RiskPaths.set.New-Data.ready.upload.log
```

Return example:

```
{
  "Status": "ready",
  "Kind": "upload",
  "ModelDigest": "d90e1e9a49a06d972ecf1d50e684c62b",
  "RunDigest": "",
  "WorksetName": "New-Data",
  "IsFolder": true,
  "Folder": "RiskPaths.set.New-Data",
  "FolderModTime": 1646803541985,
  "IsZip": true,
  "ZipFileName": "RiskPaths.set.New-Data.zip",
  "ZipModTime": 1646803541965,
  "ZipSize": 1690,
  "LogFileName": "RiskPaths.set.New-Data.ready.upload.log",
  "LogModTime": 1646803542034,
  "Lines": [
    "2022-03-09 00:25:41.964 Upload of: RiskPaths.set.New-Data ",
    "------------------ ",
    "Upload          : RiskPaths.set.New-Data.zip ",
    "Model Name      : RiskPaths ",
    "Model Version   : 3.0.0.0 2022-03-07 23:37:41.202 ",
    "Model Digest    : d90e1e9a49a06d972ecf1d50e684c62b ",
    "Scenario Name   : New-Data ",
    "Folder          : RiskPaths.set.New-Data ",
    "------------------ ",
    "2022-03-09 00:25:41.965 delete: RiskPaths.set.New-Data.ready.upload.log ",
    "2022-03-09 00:25:41.965 delete: RiskPaths.set.New-Data.error.upload.log ",
    "2022-03-09 00:25:41.965 delete: RiskPaths.set.New-Data ",
    "2022-03-09 00:25:41.966 dbcopy -m RiskPaths -dbcopy.IdOutputNames=false -dbcopy.SetName New-Data -dbcopy.To db -dbcopy.Zip -dbcopy.InputDir models\\home\\io\\uplo
ad ",
    "2022-03-09 00:25:41.983 Model RiskPaths  ",
    "2022-03-09 00:25:41.983 Unpack RiskPaths.set.New-Data.zip ",
    "2022-03-09 00:25:42.004 Workset New-Data into: 103 New-Data ",
    "2022-03-09 00:25:42.004   Parameters: 3 ",
    "2022-03-09 00:25:42.034 Done. ",
    ""
  ]
}
```

GET all upload log files for the model

GET all model uploads log files from upload directory on the server.

This method only avaliable if server configured to create uploads for user.

This is a beta version and may change in the future.

Upload can be initiated through UI or by direct API call:

- POST initiate model run upload
- POST initiate workset upload

For each upload `oms` service does create a upload log file, for example:

- `RiskPaths.set.New-Data.ready.upload.log` RiskPaths New-Data workset, upload completed and ready to use
- `RiskPaths.set.New-Data.progress.upload.log` RiskPaths New-Data, upload in progress
- `RiskPaths.set.New-Data.error.upload.log` RiskPaths New-Data, upload failed

Example of `RiskPaths.set.New-Data.ready.upload.log`:

```
2022-03-09 00:21:45.195 Upload of: RiskPaths.set.New-Data
------------------
Upload          : RiskPaths.set.New-Data.zip
Model Name      : RiskPaths
Model Version   : 3.0.0.0 2022-03-07 23:37:41.202
Model Digest    : d90e1e9a49a06d972ecf1d50e684c62b
Scenario Name   : New-Data
Folder          : RiskPaths.set.New-Data
------------------
2022-03-09 00:21:45.195 delete: RiskPaths.set.New-Data.ready.upload.log
2022-03-09 00:21:45.195 delete: RiskPaths.set.New-Data.error.upload.log
2022-03-09 00:21:45.195 delete: RiskPaths.set.New-Data
2022-03-09 00:21:45.195 dbcopy -m RiskPaths -dbcopy.IdOutputNames=false -dbcopy.SetName New-Data -dbcopy.To db -dbcopy.Zip -dbcopy.InputDir models\home\io\upload
2022-03-09 00:21:45.219 Model RiskPaths
2022-03-09 00:21:45.219 Unpack RiskPaths.set.New-Data.zip
2022-03-09 00:21:45.249 Workset New-Data into: 103 New-Data
2022-03-09 00:21:45.249   Parameters: 3
2022-03-09 00:21:45.277 Done.
```

As result of workset upload `oms` service does:

- upload archive, for example: `RiskPaths.set.New-Data.zip`
- extract workset `.csv` files with parameters into `RiskPaths.set.New-Data` folder
- extract workset `.json` metadata file
- create new or update existing New-Data workset in RiskPaths model database

Method:

```
GET /api/upload/log/model/:model
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database then result is undefined.

Call examples from browser:

```
http://localhost:4040/api/upload/log/model/RiskPaths
http://localhost:4040/api/upload/log/model/0f71660ba32bc002282c995e4552a14a
```

Return example:

```
[
  {
    "Status": "ready",
    "Kind": "upload",
    "ModelDigest": "d90e1e9a49a06d972ecf1d50e684c62b",
    "RunDigest": "",
    "WorksetName": "New-Data",
    "IsFolder": true,
    "Folder": "RiskPaths.set.New-Data",
    "FolderModTime": 1646803541985,
    "IsZip": true,
    "ZipFileName": "RiskPaths.set.New-Data.zip",
    "ZipModTime": 1646803541965,
    "ZipSize": 1690,
    "LogFileName": "RiskPaths.set.New-Data.ready.upload.log",
    "LogModTime": 1646803542034,
    "Lines": [
      "2022-03-09 00:25:41.964 Upload of: RiskPaths.set.New-Data ",
      "------------------ ",
      "Upload          : RiskPaths.set.New-Data.zip ",
      "Model Name      : RiskPaths ",
      "Model Version   : 3.0.0.0 2022-03-07 23:37:41.202 ",
      "Model Digest    : d90e1e9a49a06d972ecf1d50e684c62b ",
      "Scenario Name   : New-Data ",
      "Folder          : RiskPaths.set.New-Data ",
      "------------------ ",
      "2022-03-09 00:25:41.965 delete: RiskPaths.set.New-Data.ready.upload.log ",
      "2022-03-09 00:25:41.965 delete: RiskPaths.set.New-Data.error.upload.log ",
      "2022-03-09 00:25:41.965 delete: RiskPaths.set.New-Data ",
      "2022-03-09 00:25:41.966 dbcopy -m RiskPaths -dbcopy.IdOutputNames=false -dbcopy.SetName New-Data -dbcopy.To db -dbcopy.Zip -dbcopy.InputDir models\\home\\io\\upload ",
      "2022-03-09 00:25:41.983 Model RiskPaths  ",
      "2022-03-09 00:25:41.983 Unpack RiskPaths.set.New-Data.zip ",
      "2022-03-09 00:25:42.004 Workset New-Data into: 103 New-Data ",
      "2022-03-09 00:25:42.004   Parameters: 3 ",
      "2022-03-09 00:25:42.034 Done. ",
      ""
    ]
  },
  {
    "Status": "ready",
    "Kind": "upload",
    "ModelDigest": "d90e1e9a49a06d972ecf1d50e684c62b",
    "RunDigest": "",
    "WorksetName": "New_Scenario_of_union_duration",
    "IsFolder": true,
    "Folder": "RiskPaths.set.New_Scenario_of_union_duration",
    "FolderModTime": 1646804668330,
    "IsZip": true,
    "ZipFileName": "RiskPaths.set.New_Scenario_of_union_duration.zip",
    "ZipModTime": 1646804668308,
    "ZipSize": 2460,
    "LogFileName": "RiskPaths.set.New_Scenario_of_union_duration.ready.upload.log",
    "LogModTime": 1646804668378,
    "Lines": [
      "2022-03-09 00:44:28.308 Upload of: RiskPaths.set.New_Scenario_of_union_duration ",
      "------------------ ",
      "Upload          : RiskPaths.set.New_Scenario_of_union_duration.zip ",
      "Model Name      : RiskPaths ",
      "Model Version   : 3.0.0.0 2022-03-07 23:37:41.202 ",
      "Model Digest    : d90e1e9a49a06d972ecf1d50e684c62b ",
      "Scenario Name   : New_Scenario_of_union_duration ",
      "Folder          : RiskPaths.set.New_Scenario_of_union_duration ",
      "------------------ ",
      "2022-03-09 00:44:28.308 delete: RiskPaths.set.New_Scenario_of_union_duration.ready.upload.log ",
      "2022-03-09 00:44:28.308 delete: RiskPaths.set.New_Scenario_of_union_duration.error.upload.log ",
      "2022-03-09 00:44:28.308 delete: RiskPaths.set.New_Scenario_of_union_duration ",
      "2022-03-09 00:44:28.308 dbcopy -m RiskPaths -dbcopy.IdOutputNames=false -dbcopy.SetName New_Scenario_of_union_duration -dbcopy.To db -dbcopy.Zip -dbcopy.InputDir models\\home\\io\\upload ",
      "2022-03-09 00:44:28.328 Model RiskPaths  ",
      "2022-03-09 00:44:28.328 Unpack RiskPaths.set.New_Scenario_of_union_duration.zip ",
      "2022-03-09 00:44:28.341 Workset New_Scenario_of_union_duration into: 104 New_Scenario_of_union_duration ",
      "2022-03-09 00:44:28.341   Parameters: 3 ",
      "2022-03-09 00:44:28.378 Done. ",
      ""
    ]
  }
]
```

GET all upload log files

GET all uploads log files for from upload directory on the server.

This method only avaliable if server configured to create uploads for user.

This is a beta version and may change in the future.

Upload can be initiated through UI or by direct API call:

- POST initiate model run upload
- POST initiate workset upload

For each upload `oms` service does create a upload log file, for example:

- `RiskPaths.set.New-Data.ready.upload.log` RiskPaths New-Data workset, upload completed and ready to use
- `RiskPaths.set.New-Data.progress.upload.log` RiskPaths New-Data, upload in progress
- `RiskPaths.set.New-Data.error.upload.log` RiskPaths New-Data, upload failed

Example of `RiskPaths.set.New-Data.ready.upload.log`:

```
2022-03-09 00:21:45.195 Upload of: RiskPaths.set.New-Data
------------------
Upload          : RiskPaths.set.New-Data.zip
Model Name      : RiskPaths
Model Version   : 3.0.0.0 2022-03-07 23:37:41.202
Model Digest    : d90e1e9a49a06d972ecf1d50e684c62b
Scenario Name   : New-Data
Folder          : RiskPaths.set.New-Data
------------------
2022-03-09 00:21:45.195 delete: RiskPaths.set.New-Data.ready.upload.log
2022-03-09 00:21:45.195 delete: RiskPaths.set.New-Data.error.upload.log
2022-03-09 00:21:45.195 delete: RiskPaths.set.New-Data
2022-03-09 00:21:45.195 dbcopy -m RiskPaths -dbcopy.IdOutputNames=false -dbcopy.SetName New-Data -dbcopy.To db -dbcopy.Zip -dbcopy.InputDir models\home\io\upload
2022-03-09 00:21:45.219 Model RiskPaths
2022-03-09 00:21:45.219 Unpack RiskPaths.set.New-Data.zip
2022-03-09 00:21:45.249 Workset New-Data into: 103 New-Data
2022-03-09 00:21:45.249   Parameters: 3
2022-03-09 00:21:45.277 Done.
```

As result of workset upload `oms` service does:

- upload archive, for example: `RiskPaths.set.New-Data.zip`
- extract workset `.csv` files with parameters into `RiskPaths.set.New-Data` folder
- extract workset `.json` metadata file
- create new or update existing New-Data workset in RiskPaths model database

Method:

```
GET /api/upload/log/all
```

Call example from browser:

```
http://localhost:4040/api/upload/log/all
```

Return example:

```
[
  {
    "Status": "ready",
    "Kind": "upload",
    "ModelDigest": "ec388f9e6221e63ac248818b04633515",
    "RunDigest": "",
    "WorksetName": "Default",
    "IsFolder": true,
    "Folder": "NewCaseBased.set.Default",
    "FolderModTime": 1646804960744,
    "IsZip": true,
    "ZipFileName": "NewCaseBased.set.Default.zip",
    "ZipModTime": 1646804960719,
    "ZipSize": 1574,
    "LogFileName": "NewCaseBased.set.Default.ready.upload.log",
    "LogModTime": 1646804960806,
    "Lines": [
      "2022-03-09 00:49:20.719 Upload of: NewCaseBased.set.Default ",
      "------------------ ",
      "Upload          : NewCaseBased.set.Default.zip ",
      "Model Name      : NewCaseBased ",
      "Model Version   : 1.0.0.0 2022-03-07 23:36:46.085 ",
      "Model Digest    : ec388f9e6221e63ac248818b04633515 ",
      "Scenario Name   : Default ",
      "Folder          : NewCaseBased.set.Default ",
      "------------------ ",
      "2022-03-09 00:49:20.719 delete: NewCaseBased.set.Default.ready.upload.log ",
      "2022-03-09 00:49:20.719 delete: NewCaseBased.set.Default.error.upload.log ",
      "2022-03-09 00:49:20.719 delete: NewCaseBased.set.Default ",
      "2022-03-09 00:49:20.719 dbcopy -m NewCaseBased -dbcopy.IdOutputNames=false -dbcopy.SetName Default -dbcopy.To db -dbcopy.Zip -dbcopy.InputDir models\\home\\io\\
upload ",
      "2022-03-09 00:49:20.742 Model NewCaseBased  ",
      "2022-03-09 00:49:20.742 Unpack NewCaseBased.set.Default.zip ",
      "2022-03-09 00:49:20.767 Workset Default into: 101 Default ",
      "2022-03-09 00:49:20.767   Parameters: 3 ",
      "2022-03-09 00:49:20.806 Done. ",
      ""
    ]
  },
  {
    "Status": "ready",
    "Kind": "upload",
    "ModelDigest": "d90e1e9a49a06d972ecf1d50e684c62b",
    "RunDigest": "",
    "WorksetName": "New-Data",
    "IsFolder": true,
    "Folder": "RiskPaths.set.New-Data",
    "FolderModTime": 1646803541985,
    "IsZip": true,
    "ZipFileName": "RiskPaths.set.New-Data.zip",
    "ZipModTime": 1646803541965,
    "ZipSize": 1690,
    "LogFileName": "RiskPaths.set.New-Data.ready.upload.log",
    "LogModTime": 1646803542034,
    "Lines": [
      "2022-03-09 00:25:41.964 Upload of: RiskPaths.set.New-Data ",
      "------------------ ",
      "Upload          : RiskPaths.set.New-Data.zip ",
      "Model Name      : RiskPaths ",
      "Model Version   : 3.0.0.0 2022-03-07 23:37:41.202 ",
      "Model Digest    : d90e1e9a49a06d972ecf1d50e684c62b ",
      "Scenario Name   : New-Data ",
      "Folder          : RiskPaths.set.New-Data ",
      "------------------ ",
      "2022-03-09 00:25:41.965 delete: RiskPaths.set.New-Data.ready.upload.log ",
      "2022-03-09 00:25:41.965 delete: RiskPaths.set.New-Data.error.upload.log ",
      "2022-03-09 00:25:41.965 delete: RiskPaths.set.New-Data ",
      "2022-03-09 00:25:41.966 dbcopy -m RiskPaths -dbcopy.IdOutputNames=false -dbcopy.SetName New-Data -dbcopy.To db -dbcopy.Zip -dbcopy.InputDir models\\home\\io\\upl
oad ",
      "2022-03-09 00:25:41.983 Model RiskPaths  ",
      "2022-03-09 00:25:41.983 Unpack RiskPaths.set.New-Data.zip ",
      "2022-03-09 00:25:42.004 Workset New-Data into: 103 New-Data ",
      "2022-03-09 00:25:42.004   Parameters: 3 ",
      "2022-03-09 00:25:42.034 Done. ",
      ""
    ]
  }
]
```

GET upload files tree

GET upload files tree from upload directory on the server.

This method only avaliable if server configured to create uploads for user.

This is a beta version and may change in the future.

Upload can be initiated through UI or by direct API call:

- POST initiate model run upload
- POST initiate workset upload

As result of workset upload `oms` service does:

- upload archive, for example: `RiskPaths.set.New-Data.zip`
- extract workset `.csv` files with parameters into `RiskPaths.set.New-Data` folder
- extract workset `.json` metadata file
- create new or update existing New-Data workset in RiskPaths model database

This method retruns file tree in upload folder, for exmaple in `RiskPaths.set.New-Data` folder.

Method:

```
GET /api/upload/file-tree/:folder
```

Arguments:

```
:folder - (required) upload folder file name, for example: RiskPaths.set.New-Data
```

Call example from browser:

```
http://localhost:4040/api/upload/file-tree/RiskPaths.set.New-Data
```

Return example:

```
[
  {
    "Path": "RiskPaths.set.New-Data",
    "IsDir": true,
    "Size": 0,
    "ModTime": 1646803541985
  },
  {
    "Path": "RiskPaths.set.New-Data/RiskPaths.set.New-Data.json",
    "IsDir": false,
    "Size": 518,
    "ModTime": 1646803541984
  },
  {
    "Path": "RiskPaths.set.New-Data/set.New-Data",
    "IsDir": true,
    "Size": 0,
    "ModTime": 1646803541986
  },
  {
    "Path": "RiskPaths.set.New-Data/set.New-Data/AgeBaselinePreg1.csv",
    "IsDir": false,
    "Size": 268,
    "ModTime": 1646803541985
  },
  {
    "Path": "RiskPaths.set.New-Data/set.New-Data/SimulationCases.csv",
    "IsDir": false,
    "Size": 29,
    "ModTime": 1646803541986
  },
  {
    "Path": "RiskPaths.set.New-Data/set.New-Data/UnionStatusPreg1.csv",
    "IsDir": false,
    "Size": 199,
    "ModTime": 1646803541986
  }
]
```

POST initiate model run upload

POST model run upload request: upload model run zip file on server and start copy it into database.

This method only avaliable if server configured to create uploads for user.

This is a beta version and may change in the future.

As result of model run upload `oms` service does:

- upload archive, for example: `RiskPaths.run.New-Run.zip`
- extract `.csv` files with all run parameters and output tables into `RiskPaths.run.New-Run` folder
- extract model run metadata file: `modelOne.run.New-Run.json`
- create new or update existing "New-Run" model run in RiskPaths model database

Model run `.zip` archive must contain `.json` metadata file, which is usually created by `dbcopy` utility. For example:

```
{
    "ModelName": "RiskPaths",
    "ModelDigest": "d90e1e9a49a06d972ecf1d50e684c62b",
    "Name": "RiskPaths_Default",
    "SubCount": 1,
    "SubStarted": 1,
    "SubCompleted": 1,
    "CreateDateTime": "2022-03-22 20:49:24.341",
    "Status": "s",
    "UpdateDateTime": "2022-03-22 20:49:25.017",
    "RunDigest": "feb022eed344533046de517bddea7d09",
    "ValueDigest": "0f454b3af0d30f9f0614a9ce23e5cbfd",
    "RunStamp": "2022_03_22_20_49_24_260",
    "Txt": [{
            "LangCode": "EN",
            "Descr": "scenario",
            "Note": ""
        }, {
            "LangCode": "FR",
            "Descr": "scenario",
            "Note": ""
        }
    ],
    "Opts": {
        "OpenM.LogFilePath": "RiskPaths.log",
        "OpenM.ProgressPercent": "25",
        "OpenM.RunId": "102",
        "OpenM.RunName": "RiskPaths_Default",
        "OpenM.RunStamp": "2022_03_22_20_49_24_260",
        "OpenM.SetId": "101",
        "OpenM.SetName": "Default"
    },
    "Param": [{
            "Name": "AgeBaselineForm1",
            "Txt": [],
            "SubCount": 1,
            "DefaultSubId": 0,
            "ValueDigest": "a9a4c2d9ef657aafd89bb098635f7098"
        }, {
            "Name": "AgeBaselinePreg1",
            ........
            ........
        }
    ],
    "Table": [{
            "Name": "T01_LifeExpectancy",
            "ValueDigest": "5db49f190e7e2e999f77e1a7f796e3bc"
        }, {
            "Name": "T02_TotalPopulationByYear",
            ........
            ........
        }
    ],
    "Progress": [{
            "SubId": 0,
            "CreateDateTime": "2022-03-22 20:49:24.399",
            "Status": "s",
            "UpdateDateTime": "2022-03-22 20:49:24.818",
            "Count": 100,
            "Value": 5000
        }
    ]
}
```


Method:

```
POST /api/upload/model/:model/run
POST /api/upload/model/:model/run/:run
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:run - (optional) model run digest, run stamp or run name
```

Model run can be identified by run digest, run stamp or run name. It is recommended to use digest because it is uniquely identifies model run.

Run stamp, if not explicitly specified as model run option, automatically generated as timestamp string, ex.: 2016_08_17_21_07_55_123. It is also possible to use name, which is more human readable than digest, but if there are multiple runs with same name in database than result is undefined. This argument can committed because model run `.zip` archive must contain contain `.json` metadata file.

Call examples:

```
curl -v -X POST -F "filename=@modelOne.run.Default-4.zip" http://localhost:4040/api/upload/model/modelOne/run/Default-4
curl -v -X POST -F "filename=@modelOne.run.Default-4.zip" http://localhost:4040/api/upload/model/modelOne/run
```

POST initiate workset upload

POST model workset upload request: upload workset zip file on server and start copy it into database.

This method only avaliable if server configured to create uploads for user.

This is a beta version and may change in the future.

Workset is a set of model input parameters (a.k.a. "scenario" input). Workset can be used to run the model.

As result of workset upload `oms` service does:

- upload archive, for example: `RiskPaths.set.New-Data.zip`
- extract workset `.csv` files with parameters into `RiskPaths.set.New-Data` folder
- extract workset `.json` metadata file
- create new or update existing New-Data workset in RiskPaths model database

Workset `.zip` archive does not have to conatin workset `.json` metadata file, it can include only `.csv` files with parameter values. If workset `.zip` created by `dbcopy` utility or as result of UI download then it always contain workset `.json` metadata file, for example:

```
{
  "ModelName": "RiskPaths",
  "ModelDigest":"d90e1e9a49a06d972ecf1d50e684c62b",
  "Name": "New-Data",
  "IsReadonly": false,
  "Txt": [{
      "LangCode": "EN",
      "Descr": "Model modified set of input parameters",
      "Note": ""
    }, {
      "LangCode": "FR",
      "Descr": "Modèle modifié ensemble de paramètres d'entrée",
      "Note": "Remarques sur l'ensemble d'entrées modifiées par le modèle"
    }
  ],
  "Param": [{
      "Name": "AgeBySex",
      "SubCount": 1,
      "Txt": [{
          "LangCode": "EN",
          "Note": "Age by Sex modified values"
        }
      ]
    }
  ]
}
```

Method:

```
POST /api/upload/model/:model/workset
POST /api/upload/model/:model/workset/:set
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database than result is undefined.

```
:set - (optional) workset name
```

Workset is uniquely identified by name (inside the model). Different models can have worksets with same name, i.e. each model can have workset with name "Default". This argument is and can ommited if workset `.zip` archive contain `.json` file with workset metadata, which must include name.

Multi-part form body

Posted multi-part form consists of two parts:

- (optional) "workset-upload-options" part with JSON upload options;
- (required) workset.zip file.

JSON upload options are:

```
{
  "NoDigestCheck": true
}
```

If `NoDigestCheck` is `true` then method calls `dbcopy` with `-dbcopy.NoDigetCheck` option. By default `dbcopy` imports workset only if `ModelDigest` from source `json` metadata identical to destination model digest. If you are using `NoDigestCheck` then `dbcopy` do ignore source digest and rely on model name only.

Call examples:

```
curl -v -X POST -F "filename=@modelOne.set.New.zip" http://localhost:4040/api/upload/model/modelOne/workset/New
curl -v -X POST -F "filename=@modelOne.set.Any.zip" http://localhost:4040/api/upload/model/modelOne/workset

curl -v -X POST -F "workset-upload-options=@options.json" -F "filename=@modelOne.set.Other.zip" http://localhost:4040/api/upload/model/zz_201208171604590148_/workset
```

DELETE upload files

DELETE upload files from the server.

This method only avaliable if server configured to create uploads for user.

This is a beta version and may change in the future.

Upload can be initiated through UI or by direct API call:

- POST initiate model run upload
- POST initiate workset upload

As result of workset upload `oms` service does:

- upload archive, for example: `RiskPaths.set.New-Data.zip`
- extract workset `.csv` files with parameters into `RiskPaths.set.New-Data` folder
- extract workset `.json` metadata file
- create new or update existing New-Data workset in RiskPaths model database

This method deletes upload files from the server.

Method:

```
DELETE /api/upload/delete/:folder
```

Arguments:

```
:folder - (required) upload folder file name, for example: RiskPaths.set.New-Data
```

Call example:

```
curl -v -X DELETE http://localhost:4040/api/upload/delete/RiskPaths.set.New-Data
```

DELETE upload files asynchronously

DELETE upload files from the server asynchronously.

This method only avaliable if server configured to create uploads for user.

This is a beta version and may change in the future.

Upload can be initiated through UI or by direct API call:

- POST initiate model run upload
- POST initiate workset upload

As result of workset upload `oms` service does:

- upload archive, for example: `RiskPaths.set.New-Data.zip`
- extract workset `.csv` files with parameters into `RiskPaths.set.New-Data` folder
- extract workset `.json` metadata file
- create new or update existing New-Data workset in RiskPaths model database

This method will initiate deleting of upload files from the server. Actual delete performed in background and results can be checked through log file.

Method:

```
DELETE /api/upload/start/delete/:folder
```

Arguments:

```
:folder - (required) upload folder file name, for example: RiskPaths.set.New-Data
```

Call example:

```
curl -v -X DELETE http://localhost:4040/api/upload/start/delete/RiskPaths.set.New-Data
```

GET user views for the model

Get persistent views for the model from user home directory on the server.

This method only avaliable if server configured to save a user data in home directory.

This is a beta version and may change in the future.

Method:

```
GET /api/user/view/model/:model
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database then result is undefined.

Call examples from browser:

```
http://localhost:4040/api/user/view/model/modelOne
http://localhost:4040/api/user/view/model/a5149e422b9df4a14be0a801ec195f19
```

Return example:


```
{
  "model": {
    "name": "modelOne",
    "parameterViews": [{
        "name": "ageSex",
        "view": {
          "rows": [],
          "cols": [{
              "name": "dim1",
              "values": ["M", "F"]
            }, {
              "name": "dim0",
              "values": ["10-20", "20-30", "30-40", "40+"]
            }
          ],
          "others": [],
          "isRowColControls": true,
          "rowColMode": 2
        }
      }, {
        "name": "salaryAge",
        "view": {
          "rows": [{
              "name": "dim0",
              "values": ["L", "M", "H"]
            }, {
              "name": "dim1",
              "values": ["10-20", "20-30", "30-40", "40+"]
            }
          ],
          "cols": [],
          "others": [],
          "isRowColControls": true,
          "rowColMode": 1
        }
      }
    ]
  }
}
```

PUT user views for the model

Create new or replace existing persistent views for the model as JSON file at user home directory on the server.

This method only avaliable if server configured to save a user data in home directory.

It does update existing or save new JSON file with persistent model views in user home directory on the server.

This is a beta version and may change in the future.

Method:

```
PUT /api/user/view/model/:model
```

For example:

```
curl -v -X PUT -H "Content-Type: application/json" "http://localhost:4040/api/user/view/model/modelOne" -d @modelOne.view.json
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database then result is undefined.

JSON argument:

It is expected to be similar JSON return of GET user views for the model method.

For example (modelOne.view.json file):


```
{
  "model": {
    "name": "modelOne",
    "parameterViews": [{
        "name": "ageSex",
        "view": {
          "rows": [],
          "cols": [{
              "name": "dim1",
              "values": ["M", "F"]
            }, {
              "name": "dim0",
              "values": ["10-20", "20-30", "30-40", "40+"]
            }
          ],
          "others": [],
          "isRowColControls": true,
          "rowColMode": 2
        }
      }, {
        "name": "salaryAge",
        "view": {
          "rows": [{
              "name": "dim0",
              "values": ["L", "M", "H"]
            }, {
              "name": "dim1",
              "values": ["10-20", "20-30", "30-40", "40+"]
            }
          ],
          "cols": [],
          "others": [],
          "isRowColControls": true,
          "rowColMode": 1
        }
      }
    ]
  }
}
```


Example:

```
curl -v -X PUT -H "Content-Type: application/json" "http://localhost:4040/api/user/view/model/modelOne" -d @modelOne.view.json

*   Trying ::1...
* TCP_NODELAY set
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 4040 (#0)
> PUT /api/user/view/model/modelOne HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.55.1
> Accept: */*
> Content-Type: application/json
> Content-Length: 826
>
* upload completely sent off: 826 out of 826 bytes
< HTTP/1.1 200 OK
< Date: Tue, 20 Apr 2021 01:38:36 GMT
< Content-Length: 0
<
* Connection #0 to host localhost left intact
```

DELETE user views for the model

Delete persistent views for the model from user home directory on the server.

This method only avaliable if server configured to save a user data in home directory.

It does delete persistent user views JSON file from user home directory on the server. If such file does not exist then method does nothing and return success.

This is a beta version and may change in the future.

Method:

```
DELETE /api/user/view/model/:model
```

For example:

```
curl -v -X DELETE http://localhost:4040/api/user/view/model/modelOne
```

Arguments:

```
:model - (required) model digest or model name
```

Model can be identified by digest or by model name. It is recommended to use digest because it is uniquely identifies model. It is possible to use model name, which is more human readable than digest, but if there are multiple models with same name in database then result is undefined.

Example:

```
curl -v -X DELETE http://localhost:4040/api/user/view/model/modelOne

*   Trying ::1...
* TCP_NODELAY set
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 4040 (#0)
> DELETE /api/user/view/model/modelOne HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.55.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Location: /api/user/view/model/modelOne
< Content-Type: text/plain
< Date: Tue, 20 Apr 2021 01:41:22 GMT
< Content-Length: 0
<
* Connection #0 to host localhost left intact
```

GET web-service configuration

GET web-service configuration.

This method return web-service configuration and environment variables which names started from `OM_CFG_` prefix.

This is a beta version and may change in the future.

Method:

```
GET /api/service/config
```

Call examples:

```
http://localhost:4040/api/service/config
```

Example:


```
{
  "OmsName": "_4040",
  "RowPageMaxSize": 100,
  "RunHistoryMaxSize": 1000,
  "DoubleFmt": "%.15g",
  "LoginUrl": "",
  "LogoutUrl": "",
  "AllowUserHome": true,
  "AllowDownload": true,
  "AllowUpload": true,
  "IsJobControl": true,
  "Env": {
    "OM_CFG_INI_ALLOW": "true",
    "OM_CFG_INI_ANY_KEY": "true"
  },
  "ModelCatalog": {
    "ModelDir": "models/bin",
    "ModelLogDir": "models/log",
    "IsLogDirEnabled": true,
    "LastTimeStamp": ""
  },
  "RunCatalog": {
    "RunTemplates": [
      "run.Debug.template.txt"
    ],
    "DefaultMpiTemplate": "mpi.ModelRun.template.txt",
    "MpiTemplates": [
      "mpi.ModelRun.template.txt",
      "mpi.RiskPaths.template.txt"
    ],
    "Presets": [
      {
        "Name": "any_model.1.Use_Defaults",
        "Options": "{\r\n  \"Text\": [{\r\n      \"LangCode\": \"EN\",\r\n      \"ShortLabel\": \"Default Options\",\r\n      \"Descr\": \"Use default model run options\"\r\n    }, {\r\n      \"LangCode\": \"FR\",\r\n      \"ShortLabel\": \"Options par défaut\",\r\n      \"Descr\": \"Utiliser les options d'exécution du modèle par défaut\"\r\n    }\r\n  ],\r\n  \"subCount\": 1,\r\n  \"threadCount\": 1,\r\n  \"workDir\": \"\",\r\n  \"csvDir\": \"\",\r\n  \"csvCodeId\": \"enumCode\",\r\n  \"useIni\": false,\r\n  \"iniAnyKey\": false,\r\n  \"profile\": \"\",\r\n  \"sparseOutput\": false,\r\n  \"progressPercent\": 1,\r\n  \"progressStep\": 0.0,\r\n  \"runTmpl\": \"\",\r\n  \"mpiNpCount\": 0,\r\n  \"mpiOnRoot\": false,\r\n  \"mpiTmpl\": \"\"\r\n}\r\n"
      },
      {
        "Name": "any_model.2.Large_Run",
        "Options": "{\r\n  \"Text\": [{\r\n      \"LangCode\": \"EN\",\r\n      \"ShortLabel\": \"Large Run\",\r\n      \"Descr\": \"Large model run: use back-end MPI Cluster\"\r\n    }, {\r\n      \"LangCode\": \"FR\",\r\n      \"ShortLabel\": \"Grande Course\",\r\n      \"Descr\": \"Grande exécution de modèle : utilisez le cluster MPI back-end\"\r\n    }\r\n  ],\r\n  \"threadCount\": 3,\r\n  \"mpiNpCount\": 5,\r\n  \"mpiOnRoot\": false,\r\n  \"mpiTmpl\": \"mpi.c-all4.template.txt\"\r\n}\r\n"
      }
    ]
  }
}
```

GET web-service state

GET web-service state.

This method return web-service state, including model run state.

This is a beta version and may change in the future.

Method:

```
GET /api/service/state
```

Call examples:

```
http://localhost:4040/api/service/state
```

Example:


```
{
  "IsJobControl": true,
  "IsQueuePaused": false,
  "JobUpdateDateTime": "2022-09-13 19:51:57.436",
  "MpiRes": {
    "Cpu": 8,
    "Mem": 0
  },
  "ActiveTotalRes": {
    "Cpu": 8,
    "Mem": 0
  },
  "ActiveOwnRes": {
    "Cpu": 8,
    "Mem": 0
  },
  "QueueTotalRes": {
    "Cpu": 6,
    "Mem": 0
  },
  "QueueOwnRes": {
    "Cpu": 6,
    "Mem": 0
  },
  "MpiErrorRes": {
    "Cpu": 0,
    "Mem": 0
  },
  "LocalRes": {
    "Cpu": 4,
    "Mem": 0
  },
  "LocalActiveRes": {
    "Cpu": 0,
    "Mem": 0
  },
  "LocalQueueRes": {
    "Cpu": 0,
    "Mem": 0
  },
  "Queue": [
    {
      "SubmitStamp": "2022_09_13_19_51_25_588",
      "Pid": 0,
      "CmdPath": "",
      "ModelName": "RiskPaths",
      "ModelDigest": "d90e1e9a49a06d972ecf1d50e684c62b",
      "RunStamp": "",
      "Dir": "",
      "Opts": {
        "OpenM.BaseRunDigest": "66646f985fecfb1d59fd5ff81ee3b78a",
        "OpenM.LogRank": "true",
        "OpenM.MessageLanguage": "en-CA",
        "OpenM.NotOnRoot": "true",
        "OpenM.RunName": "RiskPaths_New-6-sub-values",
        "OpenM.SetName": "New_2022",
        "OpenM.SubValues": "6",
        "OpenM.Threads": "3"
      },
      "Env": {},
      "Threads": 3,
      "IsMpi": true,
```

```
    "Mpi": {
      "Np": 3,
      "IsNotOnRoot": true,
      "IsNotByJob": true
    },
    "Template": "mpi.ModelRun.template.txt",
    "Tables": [
      "T02_TotalPopulationByYear",
      "TG03_Union_Tables",
      "TG02_Birth_Tables"
    ],
    "RunNotes": [],
    "Res": {
      "Cpu": 6,
      "Mem": 0
    },
    "IsOverLimit": false,
    "QueuePos": 1,
    "LogFileName": "",
    "LogPath": ""
  }
],
"Active": [
  {
    "SubmitStamp": "2022_09_13_19_50_35_815",
    "Pid": 0,
    "CmdPath": "",
    "ModelName": "RiskPaths",
    "ModelDigest": "d90e1e9a49a06d972ecf1d50e684c62b",
    "RunStamp": "2022_09_13_19_51_54_081",
    "Dir": "",
    "Opts": {
      "OpenM.BaseRunDigest": "66646f985fecfb1d59fd5ff81ee3b78a",
      "OpenM.LogRank": "true",
      "OpenM.MessageLanguage": "en-CA",
      "OpenM.NotOnRoot": "true",
      "OpenM.RunName": "RiskPaths 8 subValues",
      "OpenM.SetName": "New_2022",
      "OpenM.SubValues": "8",
      "OpenM.Threads": "4"
    },
    "Env": {},
    "Threads": 4,
    "IsMpi": true,
    "Mpi": {
      "Np": 3,
      "IsNotOnRoot": true,
      "IsNotByJob": true
    },
    "Template": "mpi.ModelRun.template.txt",
    "Tables": [
      "T02_TotalPopulationByYear",
      "TG03_Union_Tables",
      "TG02_Birth_Tables"
    ],
    "RunNotes": [],
    "Res": {
      "Cpu": 8,
      "Mem": 0
    },
    "IsOverLimit": false,
    "QueuePos": 0,
    "LogFileName": "RiskPaths.2022_09_13_19_51_54_081.console.log",
    "LogPath": ""
  }
],
"History": [
  {
    "SubmitStamp": "2022_09_06_19_09_01_408",
    "ModelName": "RiskPaths",
    "ModelDigest": "d90e1e9a49a06d972ecf1d50e684c62b",
    "RunStamp": "no-run-time-stamp",
    "JobStatus": "error",
    "RunTitle": "RiskPaths_descr_tables"
  },
  {
    "SubmitStamp": "2022_09_06_23_29_01_463",
    "ModelName": "RiskPaths",
    "ModelDigest": "d90e1e9a49a06d972ecf1d50e684c62b",
    "RunStamp": "2022_09_06_23_29_05_344",
    "JobStatus": "error",
    "RunTitle": "RiskPaths_New_2022-mpi-2-descr-note"
  },
  {
    "SubmitStamp": "2022_09_06_23_30_15_733",
    "ModelName": "RiskPaths",
    "ModelDigest": "d90e1e9a49a06d972ecf1d50e684c62b",
```

```
      "RunStamp": "2022_09_06_23_30_17_893",
      "JobStatus": "success",
      "RunTitle": "RiskPaths_New_2022-mpi-2-descr-note-re-run"
    },
    {
      "SubmitStamp": "2022_09_06_23_36_48_977",
      "ModelName": "RiskPaths",
      "ModelDigest": "d90e1e9a49a06d972ecf1d50e684c62b",
      "RunStamp": "2022_09_06_23_38_38_040",
      "JobStatus": "success",
      "RunTitle": "RiskPaths next rate"
    },
    {
      "SubmitStamp": "2022_09_08_20_49_55_357",
      "ModelName": "RiskPaths",
      "ModelDigest": "d90e1e9a49a06d972ecf1d50e684c62b",
      "RunStamp": "2022_09_08_20_49_56_563",
      "JobStatus": "success",
      "RunTitle": "RiskPaths_descr_tables-2"
    }
  ],
  "ComputeState": [
    {
      "Name": "cpc-1",
      "State": "ready",
      "TotalRes": {
        "Cpu": 2,
        "Mem": 0
      },
      "UsedRes": {
        "Cpu": 2,
        "Mem": 0
      },
      "OwnRes": {
        "Cpu": 2,
        "Mem": 0
      },
      "ErrorCount": 0,
      "LastUsedTs": 1663113117436
    },
    {
      "Name": "cpc-2",
      "State": "ready",
      "TotalRes": {
        "Cpu": 2,
        "Mem": 0
      },
      "UsedRes": {
        "Cpu": 2,
        "Mem": 0
      },
      "OwnRes": {
        "Cpu": 2,
        "Mem": 0
      },
      "ErrorCount": 0,
      "LastUsedTs": 1663113117436
    },
    {
      "Name": "cpc-3",
      "State": "ready",
      "TotalRes": {
        "Cpu": 4,
        "Mem": 0
      },
      "UsedRes": {
        "Cpu": 4,
        "Mem": 0
      },
      "OwnRes": {
        "Cpu": 4,
        "Mem": 0
      },
      "ErrorCount": 0,
      "LastUsedTs": 1663113117436
    }
  ]
}
```

GET state of active model run job

GET state of active model run job.

This method allow get state of model run job which is running now, results include:

- model run request options, for example: run name, number of sub-values (sub-samples);
- model run progress and sub-values progress information;
- model run log content.

This is a beta version and may change in the future.

Method:

```
GET /api/service/job/active/:job
```

Arguments:

```
:job - (required) model run submission time stamp
```

Call examples:

```
http://localhost:4040/api/service/job/active/2022_09_13_21_28_38_409
```

Example:


```
{
  "JobStatus": "",
  "SubmitStamp": "2022_09_13_21_28_38_409",
  "Pid": 0,
  "CmdPath": "",
  "ModelName": "RiskPaths",
  "ModelDigest": "d90e1e9a49a06d972ecf1d50e684c62b",
  "RunStamp": "2022_09_13_21_30_27_952",
  "Dir": "",
  "Opts": {
    "EN.RunDescription": "Run desription in English",
    "FR.RunDescription": "Run desription in French",
    "OpenM.BaseRunDigest": "66646f985fecfb1d59fd5ff81ee3b78a",
    "OpenM.LogRank": "true",
    "OpenM.MessageLanguage": "en-CA",
    "OpenM.NotOnRoot": "true",
    "OpenM.RunName": "RiskPaths New 4 sub-values",
    "OpenM.SetName": "New_2022",
    "OpenM.SubValues": "4",
    "OpenM.Threads": "2"
  },
  "Env": {},
  "Threads": 2,
  "IsMpi": true,
  "Mpi": {
    "Np": 2,
    "IsNotOnRoot": true,
    "IsNotByJob": true
  },
  "Template": "mpi.ModelRun.template.txt",
  "Tables": [
    "T02_TotalPopulationByYear",
    "TG03_Union_Tables",
    "TG02_Birth_Tables"
  ],
  "RunNotes": [
    {
      "LangCode": "EN",
      "Note": "Run notes (English)"
    },
    {
      "LangCode": "FR",
      "Note": "Run notes (French)"
    }
  ],
  "Res": {
    "Cpu": 2,
    "Mem": 0
```

```
  },
  "IsOverLimit": false,
  "QueuePos": 0,
  "LogFileName": "RiskPaths.2022_09_13_21_30_27_952.console.log",
  "LogPath": "",
  "RunStatus": [
    {
      "ModelName": "RiskPaths",
      "ModelDigest": "d90e1e9a49a06d972ecf1d50e684c62b",
      "ModelVersion": "3.0.0.0",
      "ModelCreateDateTime": "2022-08-27 04:44:36.215",
      "Name": "RiskPaths New 4 sub-values",
      "SubCount": 4,
      "SubStarted": 4,
      "SubCompleted": 0,
      "CreateDateTime": "2022-09-13 21:30:28.188",
      "Status": "p",
      "UpdateDateTime": "2022-09-13 21:30:33.000",
      "RunDigest": "6d697389e6ca0d55b6615e02c1e453f6",
      "ValueDigest": "",
      "RunStamp": "2022_09_13_21_30_27_952",
      "Txt": [],
      "Opts": {},
      "Param": [],
      "Table": [],
      "Progress": [
        {
          "SubId": 0,
          "CreateDateTime": "2022-09-13 21:30:28.000",
          "Status": "p",
          "UpdateDateTime": "2022-09-13 21:30:33.000",
          "Count": 3,
          "Value": 92588
        },
        {
          "SubId": 1,
          "CreateDateTime": "2022-09-13 21:30:28.000",
          "Status": "p",
          "UpdateDateTime": "2022-09-13 21:30:32.000",
          "Count": 3,
          "Value": 92588
        }
      ]
    }
  ],
  "Lines": [
    "2022-09-13 21:30:28.010 RiskPaths",
    "2022-09-13 21:30:28.010 RiskPaths",
    "2022-09-13 21:30:28.030 [0] Reading C:\\go-ompp\\models\\log\\2022_09_13_21_30_27_952.run_notes.EN.md",
    "2022-09-13 21:30:28.030 [0] Reading C:\\go-ompp\\models\\log\\2022_09_13_21_30_27_952.run_notes.FR.md",
    "2022-09-13 21:30:28.101 [0] Model version  : 3.0.0.0",
    "2022-09-13 21:30:28.101 [0] Model created  : 2022-08-27 04:44:36.215",
    "2022-09-13 21:30:28.101 [0] Model digest   : d90e1e9a49a06d972ecf1d50e684c62b",
    "2022-09-13 21:30:28.101 [0] OpenM++ version: 2022-05-05 003df091e5b05b7208562c626e7dd72b4dd5055e v1.9.9",
    "2022-09-13 21:30:28.101 [0] OpenM++ build  : Windows 64 bit Release MPI",
    "2022-09-13 21:30:28.101 [0] Parallel run of 2 modeling processes, 2 thread(s) each",
    "2022-09-13 21:30:28.101 [0] OM_ROOT=C:\\go-ompp",
    "2022-09-13 21:30:28.101 [0] Model build    : Windows 64 bit Release",
    "2022-09-13 21:30:28.101 [0] Prepare fixed and missing parameters",
    "2022-09-13 21:30:28.102 [0] Run: 2022_09_13_21_30_27_952",
    "2022-09-13 21:30:28.116 [1] Model version  : 3.0.0.0",
    "2022-09-13 21:30:28.116 [1] Model created  : 2022-08-27 04:44:36.215",
    "2022-09-13 21:30:28.116 [1] Model digest   : d90e1e9a49a06d972ecf1d50e684c62b",
    "2022-09-13 21:30:28.116 [1] OpenM++ version: 2022-05-05 003df091e5b05b7208562c626e7dd72b4dd5055e v1.9.9",
    "2022-09-13 21:30:28.116 [1] OpenM++ build  : Windows 64 bit Release MPI",
    "2022-09-13 21:30:28.116 [1] OM_ROOT=C:\\go-ompp",
    "2022-09-13 21:30:28.117 [1] Model build    : Windows 64 bit Release",
    "2022-09-13 21:30:28.117 [1] Prepare fixed and missing parameters",
    "2022-09-13 21:30:28.117 [1] Run: 2022_09_13_21_30_27_952",
    "2022-09-13 21:30:28.219 [1] Run: 142 RiskPaths New 4 sub-values",
    "2022-09-13 21:30:28.219 [1] Get scenario parameters for process",
    "2022-09-13 21:30:28.220 [1] member=0 Bind scenario parameters",
    "2022-09-13 21:30:28.220 [1] member=0 Compute derived parameters",
    "2022-09-13 21:30:28.220 [1] member=1 Bind scenario parameters",
    "2022-09-13 21:30:28.221 [1] member=1 Compute derived parameters",
    "2022-09-13 21:30:28.221 [1] member=1 Prepare for simulation",
    "2022-09-13 21:30:28.221 [1] member=1 Simulation progress=0% cases=0",
    "2022-09-13 21:30:28.222 [1] member=0 Prepare for simulation",
    "2022-09-13 21:30:28.222 [1] member=0 Simulation progress=0% cases=0",
    "2022-09-13 21:30:29.687 [1] member=1 Simulation progress=1% cases=30863",
    "2022-09-13 21:30:29.849 [1] member=0 Simulation progress=1% cases=30863",
    "2022-09-13 21:30:31.153 [1] member=1 Simulation progress=2% cases=61725",
    "2022-09-13 21:30:31.483 [1] member=0 Simulation progress=2% cases=61725",
    "2022-09-13 21:30:32.618 [1] member=1 Simulation progress=3% cases=92588",
    "2022-09-13 21:30:33.139 [1] member=0 Simulation progress=3% cases=92588",
    "2022-09-13 21:30:34.089 [1] member=1 Simulation progress=4% cases=123450",
    "2022-09-13 21:30:34.781 [1] member=0 Simulation progress=4% cases=123450",
    "2022-09-13 21:30:35.550 [1] member=1 Simulation progress=5% cases=154313"
```

```
  ]
}
```

Example: empty response if model run job not found on server


```
{
  "JobStatus": "",
  "SubmitStamp": "2022_07_08_22_36_46_203",
  "Pid": 0,
  "CmdPath": "",
  "ModelName": "",
  "ModelDigest": "",
  "RunStamp": "",
  "Dir": "",
  "Opts": {},
  "Env": {},
  "Threads": 0,
  "IsMpi": false,
  "Mpi": {
    "Np": 0,
    "IsNotOnRoot": false,
    "IsNotByJob": false
  },
  "Template": "",
  "Tables": [],
  "RunNotes": [],
  "Res": {
    "Cpu": 0,
    "Mem": 0
  },
  "IsOverLimit": false,
  "QueuePos": 0,
  "LogFileName": "",
  "LogPath": "",
  "RunStatus": [],
  "Lines": []
}
```

GET state of model run job from queue

GET state of model run job from queue.

This method allow get model run job request from the queue, results include:

- model run request options, for example: run name, number of sub-values (sub-samples);

This is a beta version and may change in the future.

Method:

```
GET /api/service/job/queue/:job
```

Arguments:

```
:job - (required) model run submission time stamp
```

Call examples:

```
http://localhost:4040/api/service/job/queue/2022_09_13_21_28_38_409
```

Example:

```
{
  "JobStatus": "",
  "SubmitStamp": "2022_09_13_21_28_38_409",
  "Pid": 0,
  "CmdPath": "",
  "ModelName": "RiskPaths",
  "ModelDigest": "d90e1e9a49a06d972ecf1d50e684c62b",
  "RunStamp": "",
  "Dir": "",
  "Opts": {
    "EN.RunDescription": "Run desription in English",
    "FR.RunDescription": "Run desription in French",
    "OpenM.BaseRunDigest": "66646f985fecfb1d59fd5ff81ee3b78a",
    "OpenM.LogRank": "true",
    "OpenM.MessageLanguage": "en-CA",
    "OpenM.NotOnRoot": "true",
    "OpenM.RunName": "RiskPaths New 4 sub-values",
    "OpenM.SetName": "New_2022",
    "OpenM.SubValues": "4",
    "OpenM.Threads": "2"
  },
  "Env": {},
  "Threads": 2,
  "IsMpi": true,
  "Mpi": {
    "Np": 2,
    "IsNotOnRoot": true,
    "IsNotByJob": true
  },
  "Template": "mpi.ModelRun.template.txt",
  "Tables": [
    "T02_TotalPopulationByYear",
    "TG03_Union_Tables",
    "TG02_Birth_Tables"
  ],
  "RunNotes": [
    {
      "LangCode": "EN",
      "Note": "Run notes (English)"
    },
    {
      "LangCode": "FR",
      "Note": "Run notes (French)"
    }
  ],
  "Res": {
    "Cpu": 2,
    "Mem": 0
  },
  "IsOverLimit": false,
  "QueuePos": 0,
  "LogFileName": "",
  "LogPath": "",
  "RunStatus": [],
  "Lines": []
}
```


Example: empty response if model run job not found on server

```
{
  "JobStatus": "",
  "SubmitStamp": "2022_09_12_21_18_36_413",
  "Pid": 0,
  "CmdPath": "",
  "ModelName": "",
  "ModelDigest": "",
  "RunStamp": "",
  "Dir": "",
  "Opts": {},
  "Env": {},
  "Threads": 0,
  "IsMpi": false,
  "Mpi": {
    "Np": 0,
    "IsNotOnRoot": false,
    "IsNotByJob": false
  },
  "Template": "",
  "Tables": [],
  "RunNotes": [],
  "Res": {
    "Cpu": 0,
    "Mem": 0
  },
  "IsOverLimit": false,
  "QueuePos": 0,
  "LogFileName": "",
  "LogPath": "",
  "RunStatus": [],
  "Lines": []
}
```

GET state of model run job from history

GET state of model run job from history.

This method allow get a history of model run job, results include:

- model run request options, for example: run name, number of sub-values (sub-samples);
- model run status (success or error) and sub-values progress information;
- model run log content.

This is a beta version and may change in the future.

Method:

```
GET /api/service/job/history/:job
```

Arguments:

```
:job - (required) model run submission time stamp
```

Call examples:

```
http://localhost:4040/api/service/job/history/2022_09_13_21_28_38_409
```

Example:


```
{
  "JobStatus": "success",
  "SubmitStamp": "2022_09_13_21_28_38_409",
  "Pid": 0,
  "CmdPath": "",
  "ModelName": "RiskPaths",
  "ModelDigest": "d90e1e9a49a06d972ecf1d50e684c62b",
  "RunStamp": "2022_09_13_21_30_27_952",
  "Dir": "",
  "Opts": {
    "EN.RunDescription": "Run desription in English",
    "FR.RunDescription": "Run desription in French",
    "OpenM.BaseRunDigest": "66646f985fecfb1d59fd5ff81ee3b78a",
    "OpenM.LogRank": "true",
    "OpenM.MessageLanguage": "en-CA",
    "OpenM.NotOnRoot": "true",
    "OpenM.RunName": "RiskPaths New 4 sub-values",
    "OpenM.SetName": "New_2022",
    "OpenM.SubValues": "4",
    "OpenM.Threads": "2"
  },
  "Env": {},
  "Threads": 2,
  "IsMpi": true,
  "Mpi": {
    "Np": 2,
    "IsNotOnRoot": true,
    "IsNotByJob": true
  },
  "Template": "mpi.ModelRun.template.txt",
  "Tables": [
    "T02_TotalPopulationByYear",
    "TG03_Union_Tables",
    "TG02_Birth_Tables"
  ],
  "RunNotes": [
    {
      "LangCode": "EN",
      "Note": "Run notes (English)"
    },
    {
      "LangCode": "FR",
      "Note": "Run notes (French)"
    }
  ],
  "Res": {
    "Cpu": 2,
    "Mem": 0
```

```
},
"IsOverLimit": false,
"QueuePos": 0,
"LogFileName": "RiskPaths.2022_09_13_21_30_27_952.console.log",
"LogPath": "",
"RunStatus": [
 {
  "ModelName": "RiskPaths",
  "ModelDigest": "d90e1e9a49a06d972ecf1d50e684c62b",
  "ModelVersion": "3.0.0.0",
  "ModelCreateDateTime": "2022-08-27 04:44:36.215",
  "Name": "RiskPaths New 4 sub-values",
  "SubCount": 4,
  "SubStarted": 4,
  "SubCompleted": 4,
  "CreateDateTime": "2022-09-13 21:30:28.188",
  "Status": "s",
  "UpdateDateTime": "2022-09-13 21:35:37.090",
  "RunDigest": "6d697389e6ca0d55b6615e02c1e453f6",
  "ValueDigest": "eabadea9394ae40012fe8b70d303966e",
  "RunStamp": "2022_09_13_21_30_27_952",
  "Txt": [],
  "Opts": {},
  "Param": [],
  "Table": [],
  "Progress": [
   {
    "SubId": 0,
    "CreateDateTime": "2022-09-13 21:30:28.000",
    "Status": "s",
    "UpdateDateTime": "2022-09-13 21:33:11.000",
    "Count": 100,
    "Value": 3086250
   },
   {
    "SubId": 1,
    "CreateDateTime": "2022-09-13 21:30:28.000",
    "Status": "s",
    "UpdateDateTime": "2022-09-13 21:32:54.000",
    "Count": 100,
    "Value": 3086250
   },
   {
    "SubId": 2,
    "CreateDateTime": "2022-09-13 21:32:54.000",
    "Status": "s",
    "UpdateDateTime": "2022-09-13 21:35:18.000",
    "Count": 100,
    "Value": 3086250
   },
   {
    "SubId": 3,
    "CreateDateTime": "2022-09-13 21:33:11.000",
    "Status": "s",
    "UpdateDateTime": "2022-09-13 21:35:36.000",
    "Count": 100,
    "Value": 3086250
   }
  ]
 }
],
"Lines": [
 "2022-09-13 21:30:28.010 RiskPaths",
 "2022-09-13 21:30:28.010 RiskPaths",
 "2022-09-13 21:30:28.030 [0] Reading C:\\go-ompp\\models\\log\\2022_09_13_21_30_27_952.run_notes.EN.md",
 "2022-09-13 21:30:28.030 [0] Reading C:\\go-ompp\\models\\log\\2022_09_13_21_30_27_952.run_notes.FR.md",
 "2022-09-13 21:30:28.101 [0] Model version  : 3.0.0.0",
 "2022-09-13 21:30:28.101 [0] Model created  : 2022-08-27 04:44:36.215",
 "2022-09-13 21:30:28.101 [0] Model digest   : d90e1e9a49a06d972ecf1d50e684c62b",
 "2022-09-13 21:30:28.101 [0] OpenM++ version: 2022-05-05 003df091e5b05b7208562c626e7dd72b4dd5055e v1.9.9",
 "2022-09-13 21:30:28.101 [0] OpenM++ build  : Windows 64 bit Release MPI",
 "2022-09-13 21:30:28.101 [0] Parallel run of 2 modeling processes, 2 thread(s) each",
 "2022-09-13 21:30:28.101 [0] OM_ROOT=C:\\go-ompp",
 "2022-09-13 21:30:28.101 [0] Model build    : Windows 64 bit Release",
 "2022-09-13 21:30:28.101 [0] Prepare fixed and missing parameters",
 "2022-09-13 21:30:28.102 [0] Run: 2022_09_13_21_30_27_952",
 "2022-09-13 21:30:28.116 [1] Model version  : 3.0.0.0",
 "2022-09-13 21:30:28.116 [1] Model created  : 2022-08-27 04:44:36.215",
 "2022-09-13 21:30:28.116 [1] Model digest   : d90e1e9a49a06d972ecf1d50e684c62b",
 "2022-09-13 21:30:28.116 [1] OpenM++ version: 2022-05-05 003df091e5b05b7208562c626e7dd72b4dd5055e v1.9.9",
 "2022-09-13 21:30:28.116 [1] OpenM++ build  : Windows 64 bit Release MPI",
 "2022-09-13 21:30:28.116 [1] OM_ROOT=C:\\go-ompp",
 "2022-09-13 21:30:28.117 [1] Model build    : Windows 64 bit Release",
 "2022-09-13 21:30:28.117 [1] Prepare fixed and missing parameters",
 "2022-09-13 21:30:28.117 [1] Run: 2022_09_13_21_30_27_952",
 "2022-09-13 21:30:28.219 [1] Run: 142 RiskPaths New 4 sub-values",
 "2022-09-13 21:30:28.219 [1] Get scenario parameters for process",
 "2022-09-13 21:30:28.220 [1] member=0 Bind scenario parameters",
```

    "2022-09-13 21:30:28.220 [1] member=0 Compute derived parameters",
    "2022-09-13 21:30:28.220 [1] member=1 Bind scenario parameters",
    "2022-09-13 21:30:28.221 [1] member=1 Compute derived parameters",
    "2022-09-13 21:30:28.221 [1] member=1 Prepare for simulation",
    "2022-09-13 21:30:28.221 [1] member=1 Simulation progress=0% cases=0",
    "2022-09-13 21:30:28.222 [1] member=0 Prepare for simulation",
    "2022-09-13 21:30:28.222 [1] member=0 Simulation progress=0% cases=0",
    ....
    .... skip ....
    ....
    "2022-09-13 21:35:37.058 [0] Writing into aggregated output tables, run: 142",
    "2022-09-13 21:35:37.090 [0] Digest output tables, run: 142",
    "2022-09-13 21:35:38.251 [0] Done.",
    "2022-09-13 21:35:38.251 [1] Done."
  ]
}

Example: empty response if model run job not found on server


```
{
  "JobStatus": "",
  "SubmitStamp": "2022_07_05_19_55_38_626",
  "Pid": 0,
  "CmdPath": "",
  "ModelName": "",
  "ModelDigest": "",
  "RunStamp": "",
  "Dir": "",
  "Opts": {},
  "Env": {},
  "Threads": 0,
  "IsMpi": false,
  "Mpi": {
    "Np": 0,
    "IsNotOnRoot": false,
    "IsNotByJob": false
  },
  "Template": "",
  "Tables": [],
  "RunNotes": [],
  "Res": {
    "Cpu": 0,
    "Mem": 0
  },
  "IsOverLimit": false,
  "QueuePos": 0,
  "LogFileName": "",
  "LogPath": "",
  "RunStatus": [],
  "Lines": []
}
```

PUT model run job into other queue position

PUT model run job into other queue position.

This method is moving model run job request into other queue position, for example: move it to the top of the queue.

Only MPI cluster jobs queue can be re-ordered, using this on localhost queue does not actually change job position.

User can re-order only his own queue, other users queues are not affected. It is also impossible to move the job in front of other users job: if other user submitted model run before then it will be processed first.

This is a beta version and may change in the future.

Method:

```
PUT /api/service/job/move/:pos/:job
```

Arguments:

```
:pos - (required) new position in the queue.
```

Zero position is a top of the queue, if `pos` is negative it is treated as zero. If `pos` is greater then queue length it is terated as last position.

```
:job - (required) model run submission time stamp
```

Call examples:

```
curl -v -X PUT http://localhost:4040/api/service/job/move/99/2022_09_13_21_45_29_375
```

Example 1:

Move model run job `2022_09_13_21_45_29_375` to the top of the queue:

```
curl -v -X PUT http://localhost:4040/api/service/job/move/0/2022_09_13_21_45_29_375

*   Trying 127.0.0.1:4040...
* Connected to localhost (127.0.0.1) port 4040 (#0)
> PUT /api/service/job/move/0/2022_09_13_21_45_29_375 HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.83.1
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Content-Location: service/job/move/true/0/2022_09_13_21_45_29_375
< Content-Type: text/plain
< Date: Wed, 14 Sep 2022 01:48:02 GMT
< Content-Length: 0
<
* Connection #0 to host localhost left intact
```

Example 2:

Move model run job `2022_09_13_21_45_29_375` to the bottom of the queue using position `99999999`:

```
curl -v -X PUT http://localhost:4040/api/service/job/move/99999999/2022_09_13_21_45_29_375

*   Trying 127.0.0.1:4040...
* Connected to localhost (127.0.0.1) port 4040 (#0)
> PUT /api/service/job/move/99999999/2022_09_13_21_45_29_375 HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.83.1
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Content-Location: service/job/move/true/99999999/2022_09_13_21_45_29_375
< Content-Type: text/plain
< Date: Wed, 14 Sep 2022 01:50:22 GMT
< Content-Length: 0
<
* Connection #0 to host localhost left intact
```

DELETE state of model run job from history

DELETE state of model run job from history.

This method delete a history record of model run job. It does NOT delete actual model run data, output tables and run parameters are NOT deleted from database. Only history record of model run job is deleted.

It is a convenient method for deleting a history of failed model runs.

This is a beta version and may change in the future.

Method:

```
DELETE /api/service/job/delete/history/:job
```

Arguments:

```
:job - (required) model run submission time stamp
```

Call example:

```
curl -v -X DELETE http://localhost:4040/api/service/job/delete/history/2022_09_13_23_20_50_995

*   Trying 127.0.0.1:4040...
* Connected to localhost (127.0.0.1) port 4040 (#0)
> DELETE /api/service/job/delete/history/2022_09_13_23_20_50_995 HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.83.1
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Content-Location: /api/service/job/delete/history/2022_09_13_23_20_50_995
< Date: Wed, 14 Sep 2022 03:24:17 GMT
< Content-Length: 0
<
* Connection #0 to host localhost left intact
`
```

POST a request to refresh models catalog

On start oms web-service scan models directory tree (by default it `models/bin` directory and sub-directories) to collect all models metadata from *.sqlite database files. If we want to add, remove or overwrite model.sqlite database file(s) then it is necessary:

- close model.sqlite file(s) by POST a request to close models catalog
- refresh list of models by POST a request to refresh models catalog

This is a beta version and may change in the future.

Method:

```
POST /api/admin/all-models/refresh
```

For example:

```
curl -v -X POST http://localhost:4040/api/admin/all-models/refresh

*   Trying ::1...
* TCP_NODELAY set
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 4040 (#0)
> POST /api/admin/all-models/refresh HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.54.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Location: /api/admin/all-models/refresh/models/bin
< Date: Tue, 18 March 2019 01:02:27 GMT
< Content-Length: 0
<
* Connection #0 to host localhost left intact
```

POST a request to close models catalog

On start oms web-service scan models directory tree (by default it `models/bin` directory and sub-directories) to collect all models metadata from *.sqlite database files. If we want to add, remove or overwrite model.sqlite database file(s) then it is necessary:

- close model.sqlite file(s) by POST a request to close models catalog
- refresh list of models by POST a request to refresh models catalog

This is a beta version and may change in the future.

Method:

```
POST /api/admin/all-models/close
```

For example:

```
curl -v -X POST http://localhost:4040/api/admin/all-models/close

*   Trying ::1...
* TCP_NODELAY set
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 4040 (#0)
> POST /api/admin/all-models/close HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.54.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Location: /api/admin/all-models/close/models/bin
< Date: Tue, 18 March 2019 01:00:56 GMT
< Content-Length: 0
<
* Connection #0 to host localhost left intact
```

POST a request to pause model run queue

POST a request to pause or to resume model runs queue processing.

This is a beta version and may change in the future.

Method:

```
POST /api/admin/jobs-pause/:pause
```

Arguments:

```
:pause - (required) boolean value to pause or resume model runs queue processing.
```

It must be one of: 1, t, T, TRUE, true, True, 0, f, F, FALSE, false, False. Any other value returns an error.

Call examples:

```
curl -v -X POST http://localhost:4040/api/admin/jobs-pause/true
curl -v -X POST http://localhost:4040/api/admin/jobs-pause/1
curl -v -X POST http://localhost:4040/api/admin/jobs-pause/0
curl -v -X POST http://localhost:4040/api/admin/jobs-pause/false
```

Example:

```
curl -v -X POST http://localhost:4040/api/admin/jobs-pause/1

*   Trying 127.0.0.1:4040...
* Connected to localhost (127.0.0.1) port 4040 (#0)
> POST /api/admin/jobs-pause/1 HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.83.1
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Content-Location: /api/admin/jobs-pause/true
< Content-Type: text/plain
< Date: Fri, 22 Jul 2022 03:32:49 GMT
< Content-Length: 0
<
* Connection #0 to host localhost left intact
```

PUT a request to shutdown web-service

PUT a request to shutdown web-service.

This method shutdown web-service. It is expected to close connection and does not return any response, as result client (ex.: browser AJAX) would return an error.

This is a beta version and may change in the future.

Method:

```
PUT /api/admin/shutdown
```

Example:

```
curl -v -X PUT http://localhost:4040/api/admin/shutdown

*   Trying ::1...
* TCP_NODELAY set
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 4040 (#0)
> PUT /api/admin/shutdown HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/7.55.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Tue, 14 Apr 2020 01:33:18 GMT
< Content-Length: 18
< Content-Type: text/plain; charset=utf-8
< Connection: close
<
Shutdown completed* Closing connection 0
```