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: Use ini-files or CSV parameter files

UI: Compare model run results

UI: Aggregate and Compare Microdata

UI: Filter run results by value

UI: Disk space usage and cleanup

UI Localization: Translation of openM++

Model Development Topics

Authored Model Documentation

Built-in Attributes

Censor Event Time

Create Import Set

Derived Tables
Entity Attributes in C++
Entity Function Hooks
Entity Member Packing
Entity Tables
Enumerations
Events
Event Trace
External Names
Generated Model Documentation
Groups
Illustrative Model Align1
Lifecycle Attributes
Local Random Streams
Memory Use
Microdata Output
Model Code
Model Documentation
Model Languages
Model Localization
Model Metrics Report
Model Resource Use
Model Symbols
Parameter and Table Display and Content
Population Size and Scaling
Screened Tables
Symbol Labels and Notes
Tables
Test Models
Time-like and Event-like Attributes
Use Modules
Weighted Tabulation
File-based Parameter Values

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 R: simple loop in cloud
Run RiskPaths model from R: advanced run in cloud
Run RiskPaths model in cloud from local PC
Run model from R and save results in CSV file

Run model from R: simple loop over model parameter

Run RiskPaths model from R: advanced parameters scaling

Run model from Python: simple loop over model parameter

Run RiskPaths model from Python: advanced parameters scaling

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 output table calculated values from model run

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

Read output table values and compare model runs

Read output table values and compare model runs (enun id's)

Read microdata values from model run

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

Read aggregated microdata from model run

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

Read microdata run comparison

Read microdata run comparison (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 calculated expression(s) from model run

GET output table values and compare model runs

GET output table accumulator(s) from model run

GET output table all accumulators from model run

GET microdata values from model run

GET aggregated microdata from model run

GET microdata run comparison

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 calculated table expressions from model run

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

GET csv model runs comparison table expressions

GET csv model runs comparison table expressions (enum id's)

GET csv microdata values from model run

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

GET csv aggregated microdata from model run

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

GET csv microdata run comparison

GET csv microdata run comparison (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

POST delete multiple worksets

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

POST delete model runs

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 all download files

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 all upload files

Download and upload user files

GET user files tree

POST upload to user files

PUT create user files folder

DELETE file or folder from user files

DELETE all user files

User: manage user settings

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 service configuration

GET job service state

GET disk usage state

POST refresh disk space usage info

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 close model database

POST a request to delete the model

POST a request to open database file

POST a request to cleanup database file

GET the list of database cleanup log(s)

GET database cleanup log file(s)

POST a request to pause model run queue

POST a request to pause all model runs 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 and how to run models in cloud
- Model scripting
- 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
- How to get model gata: dbget↗
- How to copy, import or export model data: 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: Use ini-files or CSV parameter files
- UI: Compare model run results
- UI: Aggregate and Compare Microdata
- UI: Filter run results by value
- UI: Disk space usage and cleanup
- 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 how to 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 R: simple loop in cloud
- Run RiskPaths model from R: advanced run in cloud
- Run RiskPaths model in cloud from local PC
- Run model from R and save results in CSV file
- Run model from R: simple loop over model parameter
- Run RiskPaths model from R: advanced parameters scaling
- Run model from Python: simple loop over model parameter
- Run RiskPaths model from Python: advanced parameters scaling
- 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 2022
  - 'Rebuild' in Visual Studio 2022 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, 11
- Visual Studio 2022, including Community Edition, Visual Studio 2019 also works, but not tested regularly
- (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 2019 or 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 stable (12) 11 and 10, MX Linux 23, 21 and 19, Ubuntu 22.04, RedHat 9+
- 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)

Note: It does work on most of latest Linux distributions, we just not testing regularly on every Linux version.

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 20.04, Ubuntu 18.04 and RedHat 8.

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: tested on MacOS latest, may work starting from Catalina 10.15 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


```
The MIT License (MIT)

Copyright (c) 2013-present OpenM++ Contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```


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

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
- Doc folder and documentation 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 files are 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.
- `ompp_options.ompp` Model-specific source file containing `options` statements which specify commonly modified model options, such as Model Documentation options.

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]

Doc folder and documentation files

The human language documentation of an OpenM++ model can be embedded in the model source code in Label and Note comments or in files in a single model documentation folder, e.g. `RiskPaths/doc` for the `RiskPaths` model. Each model documentation file has a name and extension which determine its language and role when model documentation is built, for example:

- `NOTE.SymbolName.EN.md` Markdown file documenting the model symbol `SymbolName` in English.
- `NOTE.SymbolName.FR.md` Markdown file documenting the model symbol `SymbolName` in French.
- `LABEL.SymbolName.EN.txt` Text file containing the label of the model symbol `SymbolName` in English.
- `LABEL.SymbolName.FR.txt` Text file containing the label of the model symbol `SymbolName` in French.
- `Home.EN.md` Markdown file containing the Home topic of the Authored Model Documentation in English.
- `Home.FR.md` Markdown file containing the Home topic of the Authored Model Documentation in French.
- `TopicName.EN.md` Markdown file containing the autonomous authored topic `TopicName` in English.
- `TopicName.FR.md` Markdown file containing the autonomous authored topic `TopicName` in English.
- `*.pdf` Auxiliary downloadable PDF documentation file

See Model Documentation for more about model documentation.

[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 optional include file `omc/optional_IDE_helper.h` at the top of the model code file, as follows:

```
#include "omc/optional_IDE_helper.h" // help an IDE editor recognize model symbols
```

Modgen-specific: The optional helper include file `omc/optional_IDE_helper.h` is x-compatible and will not interfere with a Modgen build.

[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 run-time errors

Model run-time errors cause a model to emit a log message and halt with a non-zero return code. Here's an example of a model run-time error message:

```
Simulation error: attempt to assign -1 to range REPORT_TIME which has limits [0,200] when current time is 0.000000000000000 in entity_id 1 in or after event TickEvent in simulatio
n member 0 with combined seed 1
```

Model run-time errors are implemented in OpenM++ using C++ exceptions. C++ exceptions can be trapped by Visual Studio when running a Debug version of a model. If trapped, execution will halt when a run-time error occurs. This allows direct examination of entities, attributes, local variables, etc. in the Visual Studio Debugger to troubleshoot the cause of the error.

To tell Visual Studio to break if a model run-time exception occurs, ensure that `<All C++ Exceptions not in this list>` is checked in `Debug > Windows > Exception Settings`:

If a model run-time error occurs in a Visual Studio session with a Debug version of a model, display the call stack window in Visual Studio to identify the model code location which caused the error. In the following screenshot, the call stack entry `NewTimeBasedD.exe]!Ticker::TickEvent() Line 79` is the topmost model code entry in the call stack when the run-time error occurred.

Double clicking that call stack entry causes Visual Studio to navigate to the model code line which caused the error. In this example, the line is in the module `TickerCore.mpp` of the `NewCaseBased` model which was modified to deliberately cause a run-time error:

```
void Ticker::TickEvent()
{
   // Increment report time
   report_time = COERCE(REPORT_TIME, report_time + 1);
   report_time = -1;

   // Age all Persons to the current time.
...
```

In a real situation, values of attributes could be examined in the Visual Studio Debugger, exactly like debugging after a Debug break point is hit.

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...

```
{
    "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-model-ui-linux.sh",
            "problemMatcher": []
        },
        {
            "label": "start-ui-RiskPaths-release",
            "type": "shell",
            "command": "RELEASE=1 ../start-model-ui-linux.sh",
            "problemMatcher": []
        },
        {
            "label": "stop-ui-RiskPaths",
            "type": "shell",
            "command": "../stop-model-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"
        }
    ]
}
```

You also can find file above at `~/openmpp_debian_20210304/models/RiskPaths/.vscode-linux/tasks.json`

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
        }
      ]
    }
  ]
}
```

You also can find file above at `~/openmpp_debian_20210304/models/RiskPaths/.vscode-linux/launch.json`

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"
  }
}
```

You also can find file above at `~/openmpp_debian_20210304/models/RiskPaths/.vscode-linux/settings.json

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:

View Doxygen comments on hover your Model code in Visual Studio Code

It is a convenient to see Doxygen comments in your model code when you hover:

```
////////////////////////////////////////////////////////////////////////
// Event implementation

/*NOTE(Person.FirstPregEvent, EN)
    The first pregnancy event. This is the main event of analysis and
    censures
*/
```

```
TIME Person::timeFirstPregEvent()
```

Return the time to the event FirstPregEvent in the Person agent (model code).

```
TIME Person::timeFirstPregEvent()
{
    double dHazard = 0;
    TIME event_time = TIME_INFINITE;
    if (parity_status == PS_CHILDLESS)
    {
        dHazard = AgeBaselinePreg1[age_status]
          * UnionStatusPreg1[union_status];
        if (dHazard > 0)
        {
            event_time = WAIT(-log(RandUniform(1)) / dHazard);
        }
  }
  return event_time;
}
```

If such functionality does not work for you then it maybe a result of missing include path in your c++ model settings. To fix it find a missing (red underscored) include, in example below it is `#include "omc/omSimulation.h"` and select `Quick Fix` -> `Edit includePath settings`:

It should open Microsoft C/C++ extension settings page. Add `"${workspaceFolder}/../../include/**"` to your Include Path list. It is also a good idea to set C++ standard as `c++17`:

That can be done by adding `.vscode/c_cpp_properties.json` to your model folder, but such JSON maybe specific to the particular version of VSCode:

```
{
  "configurations": [
    {
      "name": "Linux",
      "includePath": [
        "${workspaceFolder}/**",
        "${workspaceFolder}/../../include/**"
      ],
      "defines": [],
      "compilerPath": "/usr/bin/gcc",
      "cStandard": "c17",
      "cppStandard": "c++17",
      "intelliSenseMode": "linux-gcc-x64"
    }
  ],
  "version": 4
}
```

You also can find file above at `~/openmpp_debian_20210304/models/RiskPaths/.vscode-linux/c_cpp_properties.json`

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
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...
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

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-model-ui-mac.sh",
            "problemMatcher": []
        },
        {
            "label": "start-ui-RiskPaths-release",
            "type": "shell",
            "command": "RELEASE=1 start-model-ui-mac.sh",
            "problemMatcher": []
        },
        {
            "label": "stop-ui-RiskPaths",
            "type": "shell",
            "command": "../stop-model-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"
        }
    ]
}
```

You also can find file above at `~/openmpp_debian_20210304/models/RiskPaths/.vscode-mac/tasks.json`

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"
    }
  ]
}
```

You also can find file above at `~/openmpp_debian_20210304/models/RiskPaths/.vscode-mac/launch.json`

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

View Doxygen comments on hover your Model code in Visual Studio Code

It is a convenient to see Doxygen comments in your model code when you hover:

```
////////////////////////////////////////////////////////////////////////
// Event implementation

/*NOTE(Person.FirstPregEvent, EN)
    The first pregnancy event. This is the main event of analysis and
    censures
*/
TIME Person::timeFirstPregEvent()
{
    double dHazard = 0;
    TIME event_time = TIME_INFINITE;
    if (parity_status == PS_CHILDLESS)
    {
        dHazard = AgeBaselinePreg1[age_status]
          * UnionStatusPreg1[union_status];
        if (dHazard > 0)
        {
            event_time = WAIT(-log(RandUniform(1)) / dHazard);
        }
  }
  return event_time;
}
```

TIME Person::timeFirstPregEvent()

Return the time to the event FirstPregEvent in the Person agent (model code).

If such functionality does not work for you then it maybe a result of missing include path in your c++ model settings. To fix it find a missing (red underscored) include, in example below it is `#include "omc/omSimulation.h"` and select `Quick Fix` -> `Edit includePath settings`:

It should open Microsoft C/C++ extension settings page. Add `"${workspaceFolder}/../../include/**"` to your Include Path list. It is also a good idea to set C++ standard as `c++17`:

That can be done by adding `.vscode/c_cpp_properties.json` to your model folder, but such JSON maybe specific to the particular version of VSCode:

```
{
  "configurations": [
    {
      "name": "Mac",
      "includePath": [
        "${workspaceFolder}/**",
        "${workspaceFolder}/../../include/**"
      ],
      "defines": [],
      "macFrameworkPath": [
        "/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX.sdk/System/Library/Frameworks"
      ],
      "compilerPath": "/usr/bin/clang",
      "cStandard": "c17",
      "cppStandard": "c++17",
      "intelliSenseMode": "macos-clang-x64"
    }
  ],
  "version": 4
}
```

You also can find file above at `~/openmpp_debian_20210304/models/RiskPaths/.vscode-mac/c_cpp_properties.json`

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: latest MacOS, may work starting from 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
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/c64da46a
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
0
0.2
0.4
0.6
0.8
1
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,∞)

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

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
- Get model gata: dbget↗
- Copy, import or export model data: dbcopy↗

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:

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

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 or .tsv file with 16 sub-values can be supplied to 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 or TSV files

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

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

If directory `C:\my_csv\` exist and contains `ParameterName.csv` or `ParameterName.tsv` file 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
```

or it can be TSV file: `C:\my_csv\Sex.tsv`:

```
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.
```

Parameters: CSV or TSV 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 or tsv 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` or `C:\my_csv\Sex.tsv`
- "Sex" parameter using sub-values [0, 15] from `C:\my_csv\Sex.csv` or `C:\my_csv\Sex.tsv`

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 dbget: Export model run results into text files

Please notice: development of `dbget` utility is not completed, additional functionality coming.

Dbget is a command line tool to export OpenM++ model metadata, input parameters and run results. It is reading from model database and produce CSV, TSV or JSON output.

Output all `modelOne` model runs input parameters and output tables into CSV or TSV files:

```
dbget -m modelOne -do all-runs
dbget -m modelOne -do all-runs -csv
dbget -m modelOne -do all-runs -tsv
```

By default dbget produces language specific output based and language is determined by matching of user OS language to model languages. Dimension labels in CSV or TSV files are language specific, for example it can be Männlich, Weiblich for Deutsche OS version. If teher is no match between user OS language and model languages in database then default model language is used. User can explicitly specify a language for `dbget` output:

```
dbget -m modelOne -do all-runs -lang FR
dbget -m modelOne -do all-runs -lang en-CA
```

Sometime you may want to have dimension item codes instead of language specific labels, e.g.: `M`, `F` instead of `Male`, `Female`. It can be achived by using `-dbget.NoLanguage` option:

```
dbget -m modelOne -do all-runs -dbget.NoLanguage
```

You can suppress zero values or NULL values in output tables:

```
dbget -m modelOne -do all-runs -dbget.NoZeroCsv
dbget -m modelOne -do all-runs -dbget.NoNullCsv
dbget -m modelOne -do all-runs -dbget.NoZeroCsv -dbget.NoNullCsv
```

To redirect CSV files into a specific directory use:

```
dbget -m modelOne -do all-runs -dir my/output/dir
```

To use `dbget` output in commands pipe do:

```
dbget -m modelOne -do all-runs -pipe
```

Important: by using `-pipe` option you suppressing all error messages. Please always check exit code to avoid errors.

If model SQLite database file located outside of current directory then you use `-db` option:

```
dbget -db /some/dir/my.sqlite -m RiskPaths -do all-runs
```

To get only a specific run results and not all model runs please specifiy model run digest, run name, run stamp or run ID or special `-dbget.LastRun`, `-dbget.FirstRun` options:

```
dbget -m modelOne -do run -dbget.FirstRun
dbget -m modelOne -do run -dbget.LastRun
dbget -m modelOne -do run -r "My special run"
dbget -m modelOne -do run -r "abcd12345efgh789eca"
dbget -m modelOne -do run -r "2024_07_16_18_34_42_038"
dbget -m modelOne -do run -dbget.RunId "1234"
```

To get only one parameter or one output table use `-parameter` or `-table` option:

```
dbget -m modelOne -dbget.LastRun -parameter ageSex
dbget -m modelOne -dbget.LastRun -table salarySex
```

It may be convinient for you to use Modgen compatibilty views to get output from openM++ model databse compatible with old Modgen tools. **Important: We are trying to mimic old Modgen database views as close as possible, however it is never be identical.*** To get Modgen comaptible model metadata:

```
dbget -m modelOne -do old-model
dbget -m modelOne -do old-model -csv
dbget -m modelOne -do old-model -tsv
dbget -m modelOne -do old-model -json
dbget -m modelOne -do old-model -pipe
```

Get model run parameters and output tables values from Modgen compatibility views:

```
dbget -m modelOne -do old-run
dbget -m modelOne -do old-run -csv
dbget -m modelOne -do old-run -tsv
dbget -m modelOne -do old-run -lang fr-CA
dbget -m modelOne -do old-run -dbget.NoLanguage
dbget -m modelOne -do old-run -pipe
dbget -m modelOne -do old-run -dir my/dir
dbget -m modelOne -do old-run -dbget.NoZeroCsv
dbget -m modelOne -do old-run -dbget.NoNullCsv
```

Get parameter run values from compatibility (Modgen) views:

```
dbget -m modelOne -do old-parameter -dbget.Parameter ageSex
dbget -m modelOne -do old-parameter -dbget.Parameter ageSex -csv
dbget -m modelOne -do old-parameter -dbget.Parameter ageSex -tsv
dbget -m modelOne -do old-parameter -dbget.Parameter ageSex -lang fr-CA
dbget -m modelOne -do old-parameter -dbget.Parameter ageSex -dbget.NoLanguage
dbget -m modelOne -do old-parameter -dbget.Parameter ageSex -pipe
```

Get output table values from compatibility (Modgen) views:

```
dbget -m modelOne -do old-table -dbget.Table salarySex
dbget -m modelOne -do old-table -dbget.Table salarySex -csv
dbget -m modelOne -do old-table -dbget.Table salarySex -tsv
dbget -m modelOne -do old-table -dbget.Table salarySex -lang fr-CA
dbget -m modelOne -do old-table -dbget.Table salarySex -dbget.NoLanguage
dbget -m modelOne -do old-table -dbget.Table salarySex -pipe
dbget -m modelOne -do old-table -dbget.Table salarySex -dbget.NoZeroCsv
dbget -m modelOne -do old-table -dbget.Table salarySex -dbget.NoNullCsv
```

There are more features avaliable in `dbget`, for example you can do microdata run comparison. And also additional functionality will be added because `dbget` is still under active development.

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

You can create .tsv output files instead of .tsv:

```
dbcopy -m modelOne -dbcopy.IntoTsv
dbcopy -m modelOne -dbcopy.IntoTsv -dbcopy.Zip
```

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.

You can create .tsv output files by using `-dbcopy.IntoTsv` option:

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

Other variation of csv or tsv output is:

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

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

You can suppress zero values and / or NULL (missing) values in output tables and microdata CSV files:

```
dbcopy -m modelOne -dbcopy.To csv -dbcopy.NoZeroCsv
dbcopy -m modelOne -dbcopy.To csv -dbcopy.NoNullCsv
dbcopy -m modelOne -dbcopy.To csv -dbcopy.NoNullCsv -dbcopy.NoZeroCsv
```

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 Any -dbcopy.ToSqlite myModel.sqlite
```

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

It is recommended to run `dbcopy -m modelOne -s modelOne_other -dbcopy.ParamDir Any` 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 Any` 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 Expressions

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 to run the model:

```
model.exe -OpenM.SubValues 16 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 or .tsv 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 or .tsv 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 or Tsv file

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

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

If directory `C:\my_csv\` exist and contains `ParameterName.csv` or `ParameterName.tsv` file model will use it parameter values. Parameter directory can be specified as command-line argument or as ini-file entry.

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 use TSV (tab separated values) files instead of CSV files, for example Sex.tsv parameter values:

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

Or even have enum id's in CSV or TSV files instead of codes, for example `C:\my_csv\Sex.id.csv` can be:

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

Model will automatically detect file format by extension:

- ParameterName.csv: CSV file with enum codes in dimension columns
- ParameterName.tsv: TSV file with enum codes in dimension columns
- ParameterName.id.csv: CSV file with enum id's in dimension columns
- ParameterName.id.tsv: TSV file with enum id's in dimension columns

You can use `OpenM.IdCsv true` model run option if your CSV or TSV files do not have .id.csv extension:

```
model.exe -OpenM.SetId 20 OpenM.IdCsv true
```

In that case model will assume parameter.csv file contains enum id's instead of enum codes.

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,param_value`, or, in case of TSV file: `sub_id dim0 dim1 param_value` header fields should be separated by TAB.

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.

It is also possible to use parameter(s) in expressions, parameter must be a scalar of float or integer type (see example of `OM_COUNT_IF` below).

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.: average value across all sub-values. To do that model developer (or model user) can specify output aggregation expression, for example, average 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_COUNT_IF(...condition...)` - count of values matching condition, `OM_COUNT_IF(acc0 > param.High)` result in SQL:

  ```
  COUNT(CASE WHEN acc0 > (....sql to select value of High parameter...) THEN 1 ELSE NULL END)
  ```

- `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 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.

Important:

- 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.

Important: Accumulators always must be inside some aggregation function, i.e. this is an error: `acc0 + OM_SUM(acc1)` because `acc0` is not aggregated. This is also an error: `OM_SUM(acc1) / OM_DIV_BY(acc0)` because `OM_DIV_BY` and `OM_IF` are non-aggregation functions. Output table expression is a single value produced as aggregation over sub-values and therefore accumulators must be wrapped into aggregation functions. For example:

- `OM_SUM( acc1 + acc0 )` is correct
- `OM_SUM( acc1 ) + acc0` incorrect
- `OM_SUM( acc2 / OM_DIV_BY(acc0) )` is correct
- `OM_SUM(acc2) / OM_DIV_BY(acc0)` incorrect
- `OM_AVG(acc2) + OM_SUM( OM_AVG(acc1) + acc0 )` is correct
- `OM_AVG(acc2) + OM_SUM( OM_AVG(acc1) ) + acc0` incorrect

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 average 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

Typically we are using SQLite database files to run the model and in that case you just specifiy a path to your sqlite file:

```
modelOne.exe -db C:\My-Model\m1.sqlite
```

It is often SQLite database file is in the same directory as model.exe file and file name is ModelName.sqlite, you can run model as:

```
my/model/dir/model.exe -OpenM.SqliteFromBin
```

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:

```
; Lines started with ; semicolon are just a comments
# Lines started with # hash are just a comments
;# You can ignore or even remove ; comments and # comments
```

```
;#==============================================================================
;#
;# 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: ModelName.sqlite
;
; Database = "Database=ModelName.sqlite; Timeout=86400; OpenMode=ReadWrite;"

;# path to SQLite database file
;#
;# If Database option (see above) specified then this SQLite option has no effect
;# Database option has higher priority over this Sqlite option.
;
; Sqlite = /path/to/my-model.sqlite

;# if SqliteFromBin is true the use model SQLite database file located next to model.exe
;# model database file path: directory/of/model/exe/ModelName.sqlite
;#
;# If any of Database or SQLite options (see above) specified then this SqliteFromBin option has no effect
;# Database and SQLite option has higher priority over this Sqlite option.
;
; SqliteFromBin = false

;# 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
;# "csv" can be any of: .csv .tsv .id.csv .id.tsv file.
;# 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
;#
;# if IdCsv is true or parameter csv file is one of: ParameterName.id.csv or ParameterName.id.tsv
;# then file contains enum id's
;# else enum code(s).
;
; 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

; OBSOLETE, for backward compatibility only
;
;# if true then use sparse output to database, default: false
;# default value: false
;# empty value:   true
;
; SparseOutput = false

; OBSOLETE, for backward compatibility only
;
;# 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]
```

```
[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
;#
;# 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
;# "csv" can be any of: .csv .tsv .id.csv .id.tsv 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
;#
;# For example of model development option processing, see function ProcessDevelopmentOptions
;# in OM_ROOT/NewCaseBased/code/ompp_framework.ompp

;#==============================================================================
;#
;# event trace model development options
;#
;# Requires activation of model development options using -OpenM.IniAnyKey (see above).
;# Requires that model code contains the statement
```

```
;#   options event_trace = on;
;# Requires OpenM.TraceToFile = true
;#   you can enable TraceToFile in section [OpenM] above, e.g.:
;#   [OpenM]
;#   TraceToFile = true
;#
;# See wiki for explanation of EventTrace options
;
; [EventTrace]
;
; format
;
; ReportStyle = readable              ; "modgen", "readable", or "csv", default: modgen
; MaximumLines = 100000               ; integer value, default: 20000
; NameColumnWidth = 20                ; integer value, default: 40
;
; filters
;
; SelectedEntityKinds = e1,e2,e3      ; comma separated list of entity kinds, if empty all entity kinds
; SelectedEntities = 1,2,3            ; comma separated list of integers, if empty all entities
; SelectLinkedEntities = no           ; default: no
; SelectedCaseSeeds = 1,2,3           ; comma separated list of case seeds, if empty all cases
; MinimumTime = 2025                  ; double value, default: -inf
; MaximumTime = 2025                  ; double value, default: +inf
; MinimumAge = 65                     ; double value, default: -inf
; MaximumAge = 66                     ; double value, default: +inf
;
; events
;
; ShowEnterSimulation = yes           ; default: yes
; ShowExitSimulation = yes            ; default: yes
; ShowEvents = yes                    ; default: yes
; SelectedEvents = e1,e2,e3           ; comma separated list of event names, if empty all events
; ShowQueuedEvents = no               ; default: no
; ShowQueuedUnchanged = no            ; default: no
; ShowSelfSchedulingEvents = no       ; default: no
; ShowQueuedSelfSchedulingEvents = no ; default: no
;
; attributes
;
; ShowAttributes = no                 ; default: no
; SelectedAttributes = year,alive     ; comma separated list of attribute names, if empty all attributes
; MinimumAttribute = 1                ; double value, default: -inf
; MaximumAttribute = 1                ; double value, default: +inf
;
; table increments
;
; ShowTableIncrements = no            ; default: no
; SelectedTables = t1,t2,t3           ; comma separated list of table names, if empty all tables

;#============================================================================
;#
;# Model run microdata: entity name and attributes to store at each model run
;#
;
[Microdata]

;# Microdata entity name and attribute names to use
;#
;# For example, if Person entity has attributes is:
;    age,  sex, income, salary, fullTime, isOldAge, pension
;# and attribute groups:
;    ageGroup, incomeGroup
;# then Person microdata attributes can be:
;
; Person = age,sex,incomeGroup
;
;# order of attributes is not important, it is defined by entity metadata and cannot be changed at run time

; 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

; Directory where microdata CSV file(s) should be created, must be existing directory
; default value: current directory
;
; CsvDir = path/to/some/directory

; 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 write event name into CSV file
;
; CsvEventColumn = true
```

OpenM++ Compiler (omc) Run 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.SqlDir` input 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`
- `-Omc.CodePage` code page for converting source files, e.g. windows-1252
- `-Omc.MessageLanguage` language to display output messages, default: user environment settings
- `-Omc.MessageFnc` localized message functions, default: `LT,logMsg,logFormatted,WriteLogEntry,WarningMsg,ModelExit`
- `-Omc.ModelDoc` generate human-readable model documentation (User Edition), default: false

- `-Omc.InDocDir` input/dir/to/find/authored/model/documentation/files/
- `-Omc.OutDocDir` output directory to create model documentation files, e.g.: `ompp/bin/doc`
- `-Omc.NoLineDirectives` suppress #line directives in generated C++, default: false
- `-Omc.NoMetadata` suppress production of model metadata (model cannot be run), default: false
- `-Omc.TraceScanning` detailed tracing from scanner, default: false
- `-Omc.TraceParsing` detailed tracing from parser, default: false
- `-OpenM.IniFile` some/optional/omc.ini

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`
- `-d` short form of `-Omc.InDocDir`
- `-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

;
; 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

;
; 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

;
; 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
```

```
; 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

; suppress #line directives in generated cpp files
; default: false
;
; NoLineDirectives = false

; dsuppress production of model metadata (model cannot be run)
; default: false
;
; NoMetadata = false

; detailed tracing from scanner
; default: false
;
; TraceScanning = false

; detailed tracing from parser
; default: false
;
; TraceParsing = false

; if true then generate model documentation
; default: false
;
; ModelDoc = false

; input directory to find authored model documentation files
; default: ../doc
;
; InDocDir = ../doc

; output directory to create model documentation files
; default: $(TARGET_DIR)/doc
;
; OutDocDir = ompp/bin/doc

;
; 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

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++:

```
[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\
       "

c-prog-win = C:\ \             ; expected result: C:\Program Files \Windows
             Program Files \ \
         Windows

; 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

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
- Compare model run results
- Aggregate and Compare Microdata
- Disk space usage and cleanup

Topic contents

- Introduction and Background
- Terminology and Concepts
- Screenshot: Chart
- Screenshot: Heat map
- Screenshot: Ad hoc measures
- 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	Measures (EN)	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: Ad hoc measures

[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

Duration of current union	Union order	First union	Second union
(-∞,1)		0.0096017	0.0370541
[1,3)		0.0199994	0.0370541
[3,5)		0.0199994	0.012775
[5,9)		0.0213172	0.012775
[9,13)		0.0150836	0.0661157
[13,∞)		0.0110791	0.0661157

[back to topic contents]

Screenshot: Session state and settings

[back to topic contents]

UI: Create new or edit 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
022-01-24 16:35:49 New scenario: a
Copy parameters from model run: Default
Find parameter...
P01_Mortality
Mortality
P02_Ferility
Fertility
P03_Unions
Union parameters
click
to add Simulation Cases parameter
from Default model run
to New_Scenario_of_union_duration
SimulationCases
Number of cases in run (over all members)
Add SimulationCases
ed
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:

Duration of current union	Union order	First union	Second union
(-∞,1)		0.0096017	0.0370541
[1,3)		0.0199994	0.0370541
[3,5)		0.0199994	0.012775
[5,9)		0.0213172	0.012775
[9,13)		0.0150836	0.0661157
[13,∞)		0.0110791	0.0661157

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.

Duration of current union	Union order	First union	Second union
(-∞,1)		0.0096017	0.0370541
[1,3)		0.0199994	0.0370541
[3,5)		0.0199994	0.012775
[5,9)		0.0213172	0.012775
[9,13)		0.0150836	0.0661157
[13,∞)		0.0110791	0.0661157

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

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

Model Runs 1 | Input Scenarios 2 | Run the Model | Downloads and Uploads

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
------------------
2022-03-08 19:54:4                                    efault.ready.up
```

Upload completed: OncoSimX-cervical.set.HPVMM_Default

UI: 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: Use ini-files or CSV parameter files

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

- UI: Run the Model
- File-based Parameter Values: CSV and TSV files
- OpenM++ Model Run Options and INI files
- UI: Upload input scenario or parameters

If user files space enabled on server *(it enabled by default in local desktop UI)* then user can:

- upload, download (view) or delete file inside of user files space
- create new or delete existing folders inside of user files space
- upload multiple files as zip archive, which would be automatically unpacked
- use uploaded INI file to run the model
- use uploaded CSV and TSV parameter files to run the model

In order to upload, download and manage user files please select your model, open `Downloads and Uploads` page and go to `Files` section at the bottom:

Click on upload button next to the folder name to upload file into that folder. You can upload zip archive with multiple files and it will be unpacked automatically.

RiskPaths: 3.0
Do full downloads, including
Upload model run .zip
Select model run .zip for u
Upload scenario .zip
Select input scenario .zip f
Ignore input scenario mod
Downloads of RiskPaths
Uploads of RiskPaths: 3
Files
Upload file
Select file t
File Upload
This PC › Local Disk (C:) › test
Organize
New folder
Program Files (x86)
ProgramData
Recovery
sqlite
SQLiteStudio
System Volume Information
Temp
test
RiskPaths_16K_cases_scenario.zip
tmp
tst
Users
util
vs_tst
VSCode
Windows
DVD Drive (D:)
RiskPaths_16K_cases_scenario.zip
RiskPaths_32K_cases.ini
SimulationSeed.csv
File name:
RiskPaths_16K_cases_scenario.zip

In order to use model run INI file or parameters CSV / TSV files please go to `Advanced Run Options` section of `Run the Model` page:

RiskPaths: 3.0.0.0: An illustrative microsimulation model of fertility beha
Models
Models list
4
Model Runs
List of model runs
1
Input Scenarios
List of input scenarios
1
Run the Model
Downloads and Uploads
View downloads and uploads
Settings
Session state and settings
Service Status
Service status and model(s) run queue
Disk Usage
View and cleanup storage space
Sub-Values (Sub-Samples):
1
Use Scenario:
Default
2024-07-26 13:48:16 Default
Use Base Run:
RiskPaths_Default
2024-07-26 13:52:20 Default firs
Default Options
Use default model run options
Large Run
Large model run: use back-end MPI
Output Tables: All
Description and Notes
Advanced Run Options
Modelling Threads max:
1
Log Progress Percent:
1
Log Progress Step:
0
Click to select model ini-file
Use INI-file:
RiskPaths.ini (Default)
Development options:
RiskPaths.ini (Default)
Click to select parameters folder with *.csv and *.tsv files
Use CSV Directory:
(None)
CSV file(s) contains Id:
(None)
Model Run Template:

Model Runs
List of model runs
Input Scenarios
List of input scenarios
Run the Model
Downloads and Uploads
View downloads and uploads
Settings
Session state and settings
Service Status
Service status and model(s) run queue
Disk Usage
View and cleanup storage space
Advanced Run Options
Modelling Threads max: 1
Log Progress Percent: 1
Log Progress Step: 0
Use INI-file: RiskPaths_32K_cases.ini
Find files...
*.ini
RiskPaths.ini (Default)
download/modelOne/run.201.Default/parameters/e.ini
2024-06-18 23:56:31.022 : 39 Bytes
RiskPaths_32K_cases.ini
2024-07-26 21:45:18.521 : 77 Bytes
some-other.ini
2024-06-20 08:35:02.340 : 118 Bytes
Development options: RiskPaths_32K_cases.ini
2.Check to use model ini-file
Uncheck if you do not want to use this model ini-file
Click on this button to see ini-file content
1.Click on file name to select this ini-file

RiskPaths: 3.0.0.0: An illustrative microsimulation model of fertility behaviour in a polit
Models
Models list
Model Runs
List of model runs
Input Scenarios
List of input scenarios
Run the Model
Downloads and Uploads
View downloads and uploads
Settings
Session state and settings
Service Status
Service status and model(s) run queue
Disk Usage
View and cleanup storage space
Modelling Threads max: 1
Log Progress Percent: 1
Log Progress Step: 0
Use INI-file: RiskPaths_32K_cases.ini
Development options: RiskPaths_32K_cases.ini
Use CSV Directory: RiskPaths_16K_cases_scenario
Find files...
*.csv *.tsv : 1 file(s)
download
2024-07-26 23:54:55.144 : 1 file(s)
RiskPaths_16K_cases_scenario
2024-07-29 20:36:33.951 : 2 file(s)
ProbMort.tsv
2024-07-29 20:36:33.950 : 1 KB
SimulationCases.csv
2024-07-29 20:36:33.951 : 30 Bytes
SimulationSeed.csv
2024-06-20 01:08:50.073 : 26 Bytes
CSV file(s) contains Id: RiskPaths_16K_cases_scenario
2.Check to use this parameters folder
Uncheck if you do not want to use this folder
1.Click on folder name to select all parameters *.csv and *.tsv files
Click on file link to see the content

UI: Compare model run results

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. If you want to compare model runs microdata please see: Aggregate and Compare Microdata

User can select multiple runs to compare results:

- select Base run first
- select one or more runs to compare (Variant runs)

Click on the Base run Info icon to view runs comparison summary:

- list of runs to Compare
- list of different parameters
- list of different output tables
- list of missing (suppressed tables)
- list of different microdata entities
- list of missing microdata entities

1-d/rp/RiskPaths: 3.0.0.0: model

scenario

```
Name:             RiskPaths_Default_4
Status:           success
Sub-values Count: 4
Started:          2023-09-26 18:21:39
Completed:        2023-09-26 18:21:39
Duration:         00:00
Run Stamp:        2023_09_26_18_21_39_141
Run Digest:       b794d3399099035740e117378c523feb
Value Digest:     e04b7489a420e0f778f53ffd2e9d1f8b
```

Model runs to compare	
New_789123_cases	New 789,123 cases
New_123_cases	New 123,456 cases
Different parameters	
SimulationCases	Number of cases in run (over all members)
Different output tables	
T01_LifeExpectancy	Life Expectancy
T02_TotalPopulationByYear	Life table
T03_FertilityByAge	Age-specific fertility
T04_FertilityRatesByAgeGroup	Fertility rates by age group
T05_CohortFertility	Cohort fertility
T06_BirthsByUnion	Pregnancies by union status & order
T07_FirstUnionFormation	First union formation

Compare output tables

Imporatnt: It is strongly recommended to drag Measure dimension on columns or rows:

Select how you want to compare run values:

- calculate values difference: `Variant - Base`
- calculate values ratio: `Variant / Base`
- calculate percentage of difference: `100* (Variant - Base) / Base`

See the relust:

- run values side by side
- calculated comparison values

It is also possible to:

- show only comparison calculated values, for example only percentage and hide source run values
- show only model run values side by side

1-d/rp/RiskPaths: 3.0.0.0: model

RiskPaths_Default_4
2023-09-26 18:21:39 scenario

New_123_cases
2023-11-04 06:22:52

Model Runs 3 | Fertility rates by age group

RiskPaths_Default_4
2023-09-26 18:21:39 scenario

T04_FertilityRat
Fertility rates by ag

show only comparison values

scenario, ...
Model run 3 / 3

[20,22.5), ...
Age interval 9 / 12

Never in union, ...
Union Status 6 / 6

PERCENT Fertility
Fertility
PERCENT Fertility

Age interval	Union Status	Measure	ew 789,123 cases	New 123,456 cases
			PERCENT Fertility	PERCENT Fertility
[20,22.5)	Never in union		100.84	100.20
	First union < 3 years		104.12	104.49
	First Union > 3 years		93.91	89.83
	After first union		250.76	268.77

1-d/rp/RiskPaths: 3.0.0.0: model

RiskPaths_Default_4
2023-09-26 18:21:39 scenario

New_123_cases
2023-11-04 06:22:52

Model Runs 3 | Fertility rates by age group

RiskPaths_Default_4
2023-09-26 18:21:39 scenario

T04_FertilityRatesByAgeGroup
Fertility rates by age group

show only Measure values side by side

scenario, ...
Model run 3 / 3

[20,22.5), ...
Age interval 9 / 12

Never in union, ...
Union Status 6 / 6

Fertility
Fertility Select Measure
PERCENT Fertility

Age interval	Union Status	Measure	enario	New 789,123 cases	New 123,456 cases
			Fertility	Fertility	Fertility
[20,22.5)	Never in union		0.054	0.055	0.055
	First union < 3 years		0.814	0.848	0.851
	First Union > 3 years		0.226	0.212	0.203
	After first union		0.023	0.056	0.061

UI: Aggregate and Compare Microdata

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. To find additional information about model run comparison please see: Compare model run results

Microdata UI has two views:

- source (raw) microdata view
- aggregation and run comparison microdata view

Age group	Gender	Salary Group	Full or part time	Is Old Age	Person entity keys	Person entity	Age of person	Income	Salary	Pension
age 40+	Male	Low	Full-time	False	1		42	28,727	21,545.25	0
age 40+	Male	Low	Full-time	False	3		44	45,109	33,831.75	0
age 40+	Male	Low	Full-time	False	5		46	61,491	46,118.25	0
age 40+	Male	Medium	Full-time	False	7		48	77,873	58,404.75	0
age 40+	Male	Medium	Full-time	False	9		50	94,255	70,691.25	0
age 40+	Female	Low	Full-time	False	0		41	20,536	15,402	0
age 40+	Female	Low	Full-time	False	2		43	36,918	27,688.5	0
age 40+	Female	Low	Full-time	False	4		45	53,300	39,975	0
age 40+	Female	Low	Full-time	False	6		47	69,682	52,261.5	0
age 40+	Female	Medium	Full-time	False	8		49	86,064	64,548	0

Aggregation and compare view allow you to aggregate source microdata attributes, group it by selected dimensions attributes and optionally compare microdata from `Base` model run to one or more `Variant` model runs.

To aggregate microdata do following:

- click on `Aggregate microdata` button to open the menu
- select desired aggregation, on screenshots example below it is `Average`
- select one or more dimension attributes to group microdata, for example: `Age Group` , `Gender`
- select one or more measure attributes, for example: `Income`
- click on `Apply` button to se the results

As result you will see an `Average` `Income` grouped by `Age Group` and `Gender`:

Compare model runs microdata

In order to compare microdata from `Base` run you need to select one or more `Variant` model runs to compare with and open your microdata. On screenshots below it is `Person` entity microdata:

After that you will need to:

- click on `Aggregate microdata` button to open the menu
- select desired comparison, on screenshots example below it is `Percent`, calculated as:

```
100 * (Variant Measure Value - Base Measure Value) / Base Measure Value
```

- select desired aggregation, for example: `Average`
- select one or more dimension attributes to group microdata, for example: `Age Group`, `Gender`
- select one or more measure attributes, for example: `Income`
- click on `Apply` button to see the results

Results below:

- Average Income grouped by Age Group and Gender side by side for Base and Variant model runs
- Calculated comparison values: Average Percent of Income difference for each Variant model runs

Important: It is strongly recommended to drag Measure dimension on columns or rows.

Important: You may need to increase "page size" (number of rows to display), default value is 10 only, below it is 40.

Adavnced microdata aggregation and comparison

If standard aggregation and comparison menu does not cover your needs then it is possible to Edit, add or remove aggregation and comparison measures.

You can use any Output Expressions but top level must be an aggregation function or combination of. For example, it can be: OM_AVG(Income) + OM_AVG(Tax) but cannot be: Income + OM_AVG(Tax).

To edit existing measures, remove or add more click on Edit... menu button:

For example to compare `Base case` and `Tax changes` then we can find how many `Person` are winners and losers in each `Age Group` and `Gender`. In order to do it:

- select `Base_values` and `TaxChanges` model runs for comparison
- in `Person` microdata `Edit...` comparison measures and enter:
  - Label `Winners 125%` is count of `Person`s who have `Tax changes` model run income > 125% of `Base case` income
  - Label `Losers 75%` is count of `Person`s who have `Tax changes` model run income < 75% of `Base case` income

```
OM_COUNT_IF( (100.0 * Income[variant] / Income[base]) > 125)
OM_COUNT_IF( (100.0 * Income[variant] / Income[base]) <  75)
```

- click on `Apply` dialog button

Person

changes	Tax changes
age Income	Average 100 * (Variant - Base) / Base Income
10,852	164
10,866	168
108,418	166
108,495	164
108,333	166
108,351	165
76,075	
74,072	

APPLY CANCEL

	Aggregation	Name	Id
	Income[variant] / Income[base]) > 125)	ex_12000	12000
	Winners: 125%		
	* Income[variant] / Income[base]) < 75)	ex_12001	12001
	Losers: 75% Aggregation expression		
+			

modelOne: 1.0: First model

Base_values
2024-04-21 16:05:38 Base case

Default
2013-05-29 23:55:07 Model One default set of parameters

Model Runs 20 | Input Scenarios 5 | Run the Model | Person entity

Base_values
2024-04-21 16:05:38 Base case

0 Size 40 Person

Base case, ...
Model run 2 / 2

Winners: 125%, ...
Person entity 2 / 2

age 10-20, ...
Age group 4 / 4

Male, ...
Gender 2 / 2

		Model run	Tax changes	Tax changes
		Person entity	Winners: 125%	Losers: 75%
Age group	Gender			
age 10-20	Male		7,968	7,200
age 10-20	Female		8,016	7,200
age 20-30	Male		39,760	36,128
age 20-30	Female		39,600	36,240
age 30-40	Male		39,840	36,240
age 30-40	Female		39,792	36,272
age 40+	Male		226,592	205,448
age 40+	Female		226,384	204,588

UI: Filter run results by value

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: Compare model run results and Aggregate and Compare Microdata.

In openM++ UI user can filter model data by selecting certain dimension items, for example, to see only `Age Group = [20, 25]` and `Province = Quebec`. It is also possible to select only data values matching certain conditions, for example, to see Persons only with `Income between 22000 and 33000`.

Below is output table `Fertility rates by age group` of RiskPaths model filtered by:

- excluding from `Age Interval` dimension first 3 age groups, where age is below 20;
- apply filter to see only data where `Fertility > 0.5`.

After user click on `Apply` button result is:

Filter conditions joined by AND and can be:

```
=      - equal to:            Age = 20
!=     - not equal to:        Age <> 20
>      - greater than:        Age > 20
>=     - greater or equal:    Age >= 20
<      - less than:           Age < 20
<=     - less or equal:       Age <= 20
IN     - in the list of:      Age IN 20, 30, 40, 50
BETWEEN - between min and max:  Age BETWEEN 30, 50
```

Numeric values has to be in "C" standard format and if you need to enter multiple values then it must be comma separated. For example: `12.5, 2345.67`, but not: `12,5; 234 5,67`.

It is also possible to fiter microdata by attribute values. For example below is Microdata compariosn:

- compare two model runs: `Default` and `New Tax Policy`
- view only `Person` microdata where `Age Group` is `age 40+`
- calculate average percent of `Income` change: `AVG( 100 * ( Income[Variant] - Income[Base] ) / Income[Base] )`
- view only data where:
  - average income increase > 130%
  - and
  - person age is one of: `44, 45, 46`

modelOne: 1.0: First model

Default
2024-06-04 14:38:15 Default

New_Tax_Policy
2024-06-04 14:40:20 New Tax Policy

Model Runs 20 | Input Scenarios 8 | Run the Model | Person entity | New_Tax_Policy | Default

Default
2024-06-04 14:38:15 Default

0 | Size 100 | [2] | Person

New Tax Policy — Model run 1 / 2

Average Age of person, ... — Person entity 2 / 6

Compare Default and New Tax Policy model runs

age 40+ — Age group 1 / 4

Male, ... — Gender 2 / 2

		Model run	New Tax Policy
		Person entity	Average 100 * (Variant - Base) / Base Income
Age group	Gender		
age 40+	Male		135
age 40+	Female		135

Average Income change: 135%

for the Person where
Income change > 130%
and
Age op Person is 44, 45, 46

APPLY

	Measure	Filter	Values
	ex_12003 Average 100 * (Variant - Base) / Base Income	>	130 Enter filter value
	ex_12000 Average Age of person	IN In the list	44, 45, 46 Enter list of comma s Enter list of comma separated values
+			

UI: Disk space usage and cleanup

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 a common practice to set a limit (quota) on storage space for cloud users. If your models data exceeded storage quota then you no longer able to run the models or upload input scenarios until cleanup is done.

In order to cleanup storage space please do:

- delete the models which you are not planning to use
- delete old files in Downloads and Uploads cloud folders
- delete old model runs
- delete old input scenarios
- reduce size of your model database file(s) as described below

It is important to remove old model runs and scenarios first and only after that click on "Cleanup Database" button. If you have not deleted anything from the model database then cleanup may not be able to reduce the size of your database file.

As first step you should close database file:

After model database is closed you can click on Cleanup database button. Cleanup may take a few minutes or multiple hours, depending on your model and number of runs remaining in the database.

You can examine database cleanup log file for details or errors:

				584 KB	2024-12-19 22:53:29.847	1-OncoSim-3.6.3.9/OncoSim.db
				60 MB	2024-12-11 01:49:20.721	modelOne.sqlite modelOne: 1.0 First model

click to see database cleanup log

	Started	Model Database	Cleanup Log File
	2024-12-19 22:53:29.104	OncoSim.sqlite	db-cleanup.2024_12_19_22_53_29_104.OncoSim.sqlite.console.txt

After cleanup is completed you need to click on Open database file button in order to use the model:

Status	Open / Close	Cleanup	Delete	Size	Updated	Model Database
				11 GB	2024-11-29 17:14:41.541	OncoSim-full/OncoSim.sqlite OncoSim: 3.6.3.9 OncoSim 3.6.3.9
				774 MB	2024-12-19 23:31:05.665	1-OncoSim-3.6.3.9/OncoSim.sqlite

click to Open model database after cleanup completed

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$/ */
```

Censor Event Time

The `censor_event_time` option enables a model-specific optimization which can reduce event queue size and improve simulation speed.

Related topics

- Model Code

Topic contents

- Introduction and Background
- Syntax and Use How to activate and use
- *Modgen-specific* Modgen issues Coding approaches for a x-compatible model

Introduction and Background

An event in an entity has an associated future time when the event will occur, provided that other intervening events do not affect that future time due to a change in attributes. If the event is represented as a hazard, code in the event time function might draw a random time to the event from an exponential distribution, like

```
tEventTime = WAIT( - log( RandUniform(1) ) / EventHazard );
```

or from some other distribution if the hazard is non-constant.

If the hazard of the event is small, the associated distribution of time-to-event will have a long tail, and the probability of drawing a time far in the future will be high. A future event time drawn from that distribution may even exceed the maximum lifespan of the entity.

If an event time exceeds the maximum lifespan of the entity, it does not need to compete with other events in the simulation because it will never occur. The event is in effect 'right-censored' by the entity's maximum lifespan.

The simulation framework arranges all the events of a model by time-to-occurrence in an event queue, and maintains that ordered queue as the simulation evolves and event times change. The work to maintain the event queue is reduced if right-censored events do not have to be inserted into their correct ordered position in the queue. Leaving out censored events also reduces the total size of the event queue, making all queue operations more efficient. Excluding right-censored events from the queue has no effect on the simulation because they are guaranteed not to occur.

Models with many rare events can gain a noticeable performance boost with event censoring. That's because a rare event occurs only rarely because its randomly drawn event time is usually far in the entity’s future. In a test with one such model, simulation time decreased by 17% with the `censor_event_time` option activated.

Independent of the `censor_event_time` option, an event with a future time of `time_infinite` will not be entered into the event queue because the event will never occur. In a sense, the `censor_event_time` option can be thought of as a generalization of this behaviour.

[back to topic contents]

Syntax and Use

The `censor_event_time` option activates the ability to specify, for each entity, a 'right-censor' time after which the entity is guaranteed to have left the simulation. By default, the option is off. To turn it on, insert the following `options` statement in a source code module:

```
options censor_event_time = on;
```

A natural place to insert this statement could be the module `ompp_framework.ompp`.

If an event time exceeds the right-censor time, the simulation framework will not insert the event into the event queue, improving efficiency. The right-censor test is redone whenever an event time changes during the simulation.

To specify the right-censor time for an entity, supply it as argument to the built-in entity function `set_censor_time` before the entity enters the simulation. If the censor time is based on other attributes such as `time` or `age`, ensure that they are assigned before the call to `set_censor_time`.

If the `censor_event_time` option is off, a call to the function `set_censor_time` has no effect.

If the `censor_event_time` option is on and the function `set_censor_time` is not called, the right-censor time is set to `time_infinite`.

Here's an example. Consider a model with two parameters, `MaxLife` and `MaxYear`. An event in the model uses `MaxLife` to stop the simulation of an entity when `age` attains the value `MaxLife`, which might, for example, be 119. Another event truncates the simulation of all entities when time attains the value `MaxYear`.

In this example, the right-censor time is set to the minimum of these two censoring events in the initialization function `Person::Start` as follows:

```
// Event time censoring optimization requires the following call
// and also activating the option censor_event_time in ompp_framework.ompp.
// It is certain that Person will leave the simulation at age MaxLife, or at year MaxYear,
// whichever comes first.
set_censor_time(std::min(WAIT(MaxLife), MaxYear));
```

The `censor_event_time` optimization is valid only if the guarantee promised by the call to `set_censor_time` is correct. A model developer can probe the correctness of the guarantee by running the model with `censor_event_time` turned off, then on, and verifying that model outputs are identical in the two runs, perhaps by using the `test_models` utility. Such a test can also measure the efficiency gain by examining the model run log files for the two runs.

[back to topic contents]

Modgen issues

A Modgen model does not contain the built-in entity function `set_censor_time`, so a Modgen build will fail with a `symbol not defined` error in the link phase of the build. This can be avoided by supplying a 'do nothing' global function with that name when building the Modgen version of a model. That can be done by inserting the following code fragment in the model source file `custom.h`:

```
#if defined(MODGEN)
// Function to set censor time in ompp.
// Supply do-nothing global function to avoid symbol not found at link stage in Modgen build.
inline void set_censor_time(double t)
{
}
#endif
```

[back to topic contents]

Create Import Set

This topic contains detailed information on the OpenM++ `create_import_set` utility. `create_import_set` creates a `zip` file suitable for upload to a downstream model, using results from an upstream model.

Topic contents

- Introduction and overview
- **Windows** Quick start
- **Linux or MacOS** Quick start
- Arguments and Options
- Worked Example: Creating an OncoSim parameter set from an HPVMM model run
- Technical Requirements: Technical information for model developers

Introduction and overview

Users familiar with linked models may wish to jump directly to the Worked Example subtopic.

A *downstream* model can use, as input, the output from a different *upstream* model. By specializing functionality in two models rather than one, a two-model design can enable analyses which would otherwise be infeasible. In particular, a time-based (interacting population) model can simulate a large population of simplified entities and feed statistical results downstream to a case-based model with more complex entities and events to simulate downstream consequences. For example, an upstream model of an infectious disease can simulate the effects of vaccination and herd immunity using an interacting population of entities to project incidence of infection over time in response to a given vaccination roll-out. That upstream model can feed aggregate results on incidence of infection over time to a more detailed downstream model with a non-interacting population to simulate health consequences, treatment, and costs.

In such a two-model design, the downstream model has input parameters whose values can be supplied by corresponding output tables from an upstream model. The pairing of output tables from the upstream model to the corresponding input parameters of the downstream model is specified by `import` statements in the source code of the downstream model. For example, the `import` statement

```
import IncidenceRatesHPV (HPVMM.IM_Incidence) sample_dimension=on;
```

in a downstream model (in this example, a model named OncoSim) specifies that the input parameter `IncidenceRatesHPV` of OncoSim can be provided by the output table `IM_Incidence` of the upstream model HPVMM.

Multiple output tables from an upstream model can supply values to multiple parameters in the downstream model, with each such linkage specified by an `import` statement in the downstream model source code. For logical coherence, every such linked table from the *same run* of the upstream model needs to be used as input in the corresponding parameter in the downstream model. For example, the upstream model HPVMM supplies 11 output tables which match 11 input parameters in the downstream model OncoSim. It is essential that each of the 11 output tables from the *same* HPVMM run be copied to the corresponding input parameter for an OncoSim run.

The `create_import_set` utility supports the propagation of results from an upstream model to a downstream model by building a partial parameter set for the downstream model from a run of the upstream model. A user 1) downloads a run from the upstream model to their workstation, 2) runs the `create_import_set` utility, and finally 3) uploads the resulting set for use by the downstream model. Once uploaded, the set can be used to construct one or more scenarios for the downstream model.

[back to topic contents]

Windows Quick Start

Verify installation of `create_import_set` (Windows)

A Windows executable version of `create_import_set` is distributed with OpenM++ at `OM_ROOT/bin/create_import_set.exe`, where `OM_ROOT` stands for the OpenM++ installation directory.

To test installation and operation of `create_import_set`, open a command prompt, change the current directory to `OM_ROOT/bin`, and type the command

`create_import_set -v`

Output should be similar to the following:

`create_import_set version 1.0`

`create_import_set` is written in the Perl language, and distributed with OpenM++ at `OM_ROOT/Perl/create_import_set.pl`. Examples in this topic may invoke `test_models` using the Perl interpreter from the `OM_ROOT/Perl` directory, for example

`perl create_import_set.pl -v`

On Windows, unless you have Perl and the required Perl components installed, invoke the executable version of `create_import_set` from the `OM_ROOT/bin` directory with a command like

`create_import_set -v`

[back to topic contents]

Linux or MacOS Quick Start

Verify installation of `create_import_set` (Linux, MacOS)

`create_import_set` is a Perl script distributed with OpenM++ at `OM_ROOT/Perl/create_import_set.pl`, where `OM_ROOT` stands for the OpenM++ installation directory. To test installation and operation of `create_import_set`, open a command prompt, change the current directory to `OM_ROOT\Perl`, and type the command

`perl create_import_set.pl -v`

Output should be similar to the following:

`create_import_set version 1.0`

Depending on your operating system version and installation history, Perl may ask you to install missing Perl modules required by `create_import_set.pl`. If so, it will name them explicitly when you invoke `create_import_set.pl`. We do recommend to use `cpanm` for Perl modules installation. Typical scenario is:

```
cpan App::cpanminus             # initialize cpanm, if not done before

cpanm Getopt::Long::Descriptive
cpanm Capture::Tiny
cpanm Config::Tiny
cpanm File::Copy::Recursive
cpanm File::Which
```

Above list of modules can be different and depends on your current Perl configuration, and on the version of create_import_set.

[back to topic contents]

Arguments and Options

This subtopic describes the command line options and arguments of `create_import_set`.

A complete list of options is displayed by issuing the command

`perl create_import_set.pl -h`

Output should be similar to the following:

```
create_import_set [-dhiruvw] [long options...]
        -h --help                   print usage message and exit
        -v --version                print version and exit
        -i STR --imports STR        path of model imports csv file
        -u STR --upstream STR       name of upstream model
        -r STR --run STR            name of upstream model run
        -d STR --downstream STR     name of downstream model
        -w STR --workdir STR        path of working directory for zips
                                    (default is current directory)
        --keep                      keep and propagate all subs
        --verbose                   verbose log output
```

A value is required for the following arguments:

- -i imports file: The model imports csv file for the downstream model
- -u upstream model name: The name of the upstream model
- -r upstream run name: The name of the upstream model run
- -d downstream model name: The name of the downstream model

Each of these 4 arguments is described below. Or jump to the Worked Example for a concrete illustration.

The `-w` argument is optional. It specifies the directory where the input zip downloaded from the upstream model run is found. It is also the directory where the output zip for the downstream model will be constructed. By default, the working directory is the current working directory of the terminal session in which `create_import_set` is invoked.

The `--keep` option is an experimental option intended for linked models which use OpenM++ architecture for parameter uncertainty. It is not compatible with linked models in Modgen.

The `--verbose` option outputs detailed diagnostics of no interest to an end user.

-i imports file

The model imports `csv` file contains information about the pairing of tables in an upstream model with the parameters in a downstream model. It has a name of the form `MODEL.imports.csv` where `MODEL` is the name of the downstream model. This file contains all import information for the downstream model, which might include imports from multiple upstream models. For example, a downstream model OncoSimX might import parameters from an upstream model HPVMM as well as an upstream model GMM. The downstream model developer can provide a copy of this file. The file is generated by the OpenM++ compiler when the downstream model is built. It is located in the output `src` directory in the model build directory structure.

[back to arguments and options]
[back to topic contents]

-u upstream model name

A downstream model can be linked to more than one upstream model. The `-u` option specifies which upstream model is to be used by `create_import_set` to create the parameter set for the downstream model. Valid model names are in a column of the model imports csv file and come from the source code of the downstream model.

[back to arguments and options]
[back to topic contents]

-r upstream run name

The tables from an upstream model run are in a zip file downloaded previously using the OpenM++ UI. The name of that zip file is constructed automatically from the model name and run name when the run is downloaded, for example `HPVMM.run.DefaultVaccination.zip` for a run named `DefaultVaccination` of the `HPVMM` model.

[back to arguments and options]
[back to topic contents]

-d downstream model name

The name of the downstream model is required and must match the name of a model in the target OpenM++ database. `create_import_set` will construct a partial parameter set ready for upload using this name and the name of the upstream model run, for example `OncoSimX.set.DefaultVaccination.zip` for `-d OncoSimX`. Note that the partial parameter set for the downstream model has the same name as the upstream model run, `DefaultVaccination` in this example.

[back to arguments and options]
[back to topic contents]

Worked Example

This worked example, in Windows, uses an upstream model named HPVMM and a downstream model named OncoSimX. Note that these models are not distributed with OpenM++. The example assumes a remote server houses an instance of the upstream and downstream models, but works equally well for models on a workstation. This example could also have been done without the OpenM++ UI by using the `dbcopy` utility.

Step 0a: Create a staging directory on your workstation to manage downloaded runs and construct parameter sets for upload, for example `C:\Analysis\runs`.

Step 0b: Get a copy of the `create_import_set` utility on your workstation. For Windows users, a stand-alone executable version can be found in the OpenM++ distribution in the `bin` sub-folder. The Perl version is located in the `perl` sub-folder. If you use OpenM++ on your desktop for model development, you can invoke the utility using the environment variable `OM_ROOT`, e.g. `%OM_ROOT%\bin\create_import_set`. If you do not use OpenM++ for development, you may find it more convenient to just copy the utility executable to your staging directory.

Step 0c: Copy the imports file for the downstream model to the staging directory. In this example, the imports file for OncoSimX is named `OncoSimX.imports.csv`. The file is generated by the OpenM++ compiler in a model build folder named `OncoSimX\ompp\src`. Ask the model developer for a copy of this file if you don't build the model yourself.

Step 1: Do a run of the upstream model (HPVMM in this example), and give it a short but meaningful name. In this example, the run is named `DefaultVaccination`.

Step 2: In the UI, make sure that the option `Full, compatible with desktop model` is checked in the `Model Downloads` section of the options panel.

Step 3: In the UI, select the run and click the download button. Wait for the server to construct the download zip for the run, then click it to download it to the downloads folder on your workstation. In this example, the file is named `HPVMM.run.DefaultVaccination.zip`. Copy or move the file to the staging directory.

Step 4: Open a command prompt and navigate to the staging directory in the terminal window.

Step 5: Issue the command

```
create_import_set -i OncoSimX.imports.csv -u HPVMM -d OncoSimX -r DefaultVaccination
```

The arguments name the upstream model with `-u`, the downstream model with `-d`, the name of the upstream run with `-r`, and specify the imports file `OncoSimX.imports.csv` which `create_import_set` uses to identify and transform each imported upstream table to the corresponding downstream parameter. `create_input_set` uses the arguments to construct the names of the input and output zip files.

The utility may take 10 or more seconds to run. When it completes, it writes something like the following to the terminal window to indicate success.

```
11 downstream OncoSimX parameters created from upstream HPVMM tables
```

If you examine the staging directory, you'll notice a new file named `OncoSimX.set.DefaultVaccination.zip`. It contains the 11 OncoSimX parameters with values from the 11 corresponding HPVMM tables in the HPVMM run named `DefaultVaccination`.

Step 6: Use the UI to upload the set to the server.

The set is uploaded as a read-only partial scenario. You can use the UI to combine it with a previous OncoSim run or scenario to incorporate the results of the HPVMM `DefaultVaccination` run.

To make clear the provenance of downstream parameters, `create_import_set` generates, for each imported parameter, a parameter value description indicating the upstream model name and run name, e.g. `HPVMM: DefaultVaccination`. That description will follow each generated parameter value in any downstream OncoSimX run which uses, directly or indirectly, the partial parameter set generated by `create_import_set`.

HPVMM-1.9.1.0/HPVMM: 1.9.1.0: HPVMM 1.9.1.0
Models
Models list
15
Model Runs
List of model runs
2
Input Scenarios
List of input scenarios
2
Run the Model
Downloads and Uploads
View downloads and uploads
Settings
Session state and settings
Model Runs 2
Input Scenarios 2
Run the Model
Downloads and U
2022-06-03 14:29:56.122
Ready
In progress
Failed
Total
Downloads
2
0
0
2
Uploads
0
0
0
0
Do full downloads, compatible with desktop model
Do fast downloads, only to analyze output values
Upload model run .zip
before doing
any downloads select
"Do full downloads"
option
Command Prompt
Microsoft Windows [Version 10.0.19044.1706]
(c) Microsoft Corporation. All rights reserved.
C:\>cd CPAC\HPVMM
C:\CPAC\HPVMM>\openmpp_win_20220505\bin\create_import_set -i OncoSimX-cervical.imports.csv -u HPVMM -r HPVMM_90 -d OncoSimX-cervical
11 downstream OncoSimX-cervical parameters created from upstream HPVMM tables
C:\CPAC\HPVMM>

HPVMM: HPVMM 1.9.1.0
Model Runs 1
Input Scenarios 1
Run the Model
Parameters 11
Output Tables 5
HPVMM_Default
2022-03-08 16:29:36 First model r
Find model run...
1. download HPVMM model run results
HPVMM_Default
2022-03-08 16:29:36 First model run with Default input scenar

[back to topic contents]

Technical Requirements

The following restrictions apply:

The input run zip must contain only a single run.

The classification levels in each parameter / table pair must match, for each dimension.

The upstream table dimensions must be named `Dim0`, `Dim1`, ...

Each imported table can contain only a single measure named `Value`.

The downstream target parameter dimensions must be named `Dim0`, `Dim1`, ...

For robustness, provide explicit names to dimensions in the upstream and downstream models to respect the name requirements. For example, the upstream model could have a source code module `ExplicitNames.ompp` with content like:

```
// Table IM_ClearanceHazard: Inter-model: Clearance hazard
//NAME IM_ClearanceHazard.Dim0 Dim0
//NAME IM_ClearanceHazard.Dim1 Dim1
//NAME IM_ClearanceHazard.VALUE Value

// Table IM_PersistentProportion: Inter-model: Persistent proportion
//NAME IM_PersistentProportion.Dim0 Dim0
//NAME IM_PersistentProportion.VALUE Value

// Table IM_Incidence: Inter-model: Incidence
//NAME IM_Incidence.Dim0 Dim0
//NAME IM_Incidence.Dim1 Dim1
//NAME IM_Incidence.Dim2 Dim2
//NAME IM_Incidence.Dim3 Dim3
//NAME IM_Incidence.Dim4 Dim4
//NAME IM_Incidence.Dim5 Dim5
//NAME IM_Incidence.VALUE Value
```

and the downstream model could have a source code module named `ExplicitNames.ompp` with content like:

```
// Parameter HpvClearanceHazard: HPV infection clearance rates
//   from HPVMM Table IM_ClearanceHazard: Inter-model: Clearance hazard
//   leading dimension Dim0 is parameter set
//NAME HpvClearanceHazard.Dim0 Dim0
//NAME HpvClearanceHazard.Dim1 Dim1
//NAME HpvClearanceHazard.Dim2 Dim2

// Parameter HpvPersistentProportion: Proportion who cannot naturally clear HPV infection
//   from HPVMM Table IM_PersistentProportion: Inter-model: Persistent proportion
//   leading dimension Dim0 is parameter set
//NAME HpvPersistentProportion.Dim0 Dim0
//NAME HpvPersistentProportion.Dim1 Dim1

// Parameter IncidenceRatesHPV: Incidence rates of HPV
//   from HPVMM Table IM_Incidence: Inter-model: Incidence
//   leading dimension Dim0 is parameter set
//NAME IncidenceRatesHPV.Dim0 Dim0
//NAME IncidenceRatesHPV.Dim1 Dim1
//NAME IncidenceRatesHPV.Dim2 Dim2
//NAME IncidenceRatesHPV.Dim3 Dim3
//NAME IncidenceRatesHPV.Dim4 Dim4
//NAME IncidenceRatesHPV.Dim5 Dim5
//NAME IncidenceRatesHPV.Dim6 Dim6
```

[back to topic contents]

Derived Tables

This topic is under construction and/or revision.

Topic summary, two sentences max.

Related topics

- Model Code

Topic contents

- Introduction and outline

Introduction and outline

Content to follow.

[back to topic contents]

Entity Attributes in C++

This topic contains detailed information about specific C++ compiler errors you may encounter in C++ model code which uses entity attributes, eg the `age` attribute of a `Person` entity. This topic describes the situations which can cause these specific C++ error messages and how to address them in model code.

Topic contents

- Introduction
- Ternary operator
- Attribute assignment
- min, max, and clamp

Introduction

Forthcoming content

[back to topic contents]

Ternary operator

Forthcoming content

[back to topic contents]

Attribute assignment

Forthcoming content

[back to topic contents]

min, max, and clamp

Forthcoming content

[back to topic contents]

Entity Function Hooks

This topic describes the `hook` statement in model source code. It is used to chain the execution of one entity function to another entity function.

Related topics

- Model Code

Topic contents

- Introduction and Motivation
- Syntax and Use Syntax and example
- Disambiguating Hook Order When hooks collide

Introduction and Motivation

Content to follow.

Content sketch:

- Increase modularity
- Reduce model code 'busy work'

[back to topic contents]

Syntax and Use

A `hook` statement is specified within an entity declaration, and has the form

```
hook downstream_function, upstream_function, order;
```

The integer argument `order` is optional and is implicitly `0` if not specified.

In the following example, the downstream function `StartCity` is called by the upstream function `Start'.

```
entity Person
{
   void StartCity();
   hook StartCity, Start;
};
```

If an upstream function `F` has been hooked to by one or more downstream functions, the model developer must insert a call to a function named `hook_F` in the definition of the upstream function `F`. In the following example, the model developer inserted a call to the function `hook_Start` in an appropriate place in the definition of the function `Start`. The body of the function `hook_Start` is generated by the OpenM++ compiler, and calls, in hook order, all functions hooked to the `Start` function.

```
void Person::Start()
{
   // Initialize all attributes.
   initialize_attributes();
...
   hook_Start();

   // Have the entity enter the simulation.
   enter_simulation();
}
```

The OpenM++ compiler will raise an error if the body of a function contains no invocation of the `hook_F` function if one or more downstream functions hook to it.

[back to topic contents]

Disambiguating Hook Order

For the model source code to be logically well-specified, the order of invocation of multiple hooks to the same function must be specified.

For example, if the `TestEventMemory.mpp` module contains

```
actor Person
{
   //EN Start city
   void StartCity();
   hook StartCity, Start;
};
```

and the `TestFixed.mpp` module contains

```
actor Person
{
   //EN Set dog ownership
   void StartDogOwnership();
   hook StartDogOwnership, Start;
};
```

The OpenM++ compiler will issue two warnings like

```
TestEventMemory.mpp(93): warning : one or more functions hooking to 'Start' are ordered ambiguously with respect to 'StartCity'.
TestFixed.mpp(48): warning : one or more functions hooking to 'Start' are ordered ambiguously with respect to 'StartDogOwnership'.
```

The warning is issued for each `hook` statement which has an order tied to another hook statement. In this example, the two `hook` statements did not specify hook order, so they both had implicit order `0`, creating ambiguity. The module and line number of these warnings are the code locations of the `hook` statements responsible for the ambiguity. In an IDE, the warning can be clicked to navigate directly to the `hook` statements responsible for the ambiguity.

To resolve the ambiguity, supply an explicit order to the `hook` which does not conflict, for example

```
   hook StartCity, Start, 1;
   hook StartDogOwnership, Start, 2;
```

[back to topic contents]

Entity Member Packing

This topic describes how to pack entity members to reduce memory use.

Related topics

- Model Code
- Model Resource Use

Topic contents

- Introduction and Background
- Syntax and Use How to activate and use

Introduction and Background

The `entity_member_packing` option can reduce the size of entities with no impact on processing performance. It can be useful for time-based models with large populations. The option does negate a convenience feature for model debugging (see below).

An entity is implemented as a C++ object with data members consisting of the entity's attributes and other data members generated by the OpenM++ compiler. Each entity in a run occupies a block of contiguous memory containing the values of its data members. For example, each `Person` entity in the RiskPaths model has a block of 776 bytes of memory containing 70 data members (38 attributes and 32 other data members).

Each data member has a size in bytes determined by its type. For example, in the RiskPaths model the value of the built-in attribute `age` is held in a C++ `double` which is 8 bytes in size. The attribute `in_union` is held in a C++ `bool` which is 1 byte in size. The attribute `unions` is held in a C++ `int` which is 4 bytes in size. Below is an extract of a table produced by the `resource_use` option which lists each data member of the `Person` entity in RiskPaths and its size in bytes.

```
+-----------------------------------------+
|     Person Members (detail)             |
+----------------------------------+-------+
| member                           | bytes |
+----------------------------------+-------+
| Attributes:                      |       |
|   Built-in:                      |       |
|     age                          |     8 |
|     case_id                      |     8 |
|     case_seed                    |     8 |
|     entity_id                    |     4 |
|     events                       |     4 |
|     time                         |     8 |
|   Simple:                        |       |
|     life_status                  |     1 |
|     parity_status                |     1 |
|     union_period2_change         |     8 |
|     union_status                 |     1 |
|     unions                       |     4 |
|   Maintained:                    |       |
|     age_status                   |     4 |
|     dissolution_duration         |     4 |
|     dissolution_hazard           |     8 |
|     formation_hazard             |     8 |
|     in_union                     |     1 |

...

+----------------------------------+-------+
| Sum of member bytes              |   699 |
| Bytes per entity                 |   776 |
| Storage efficiency (%)           |  90.1 |
+----------------------------------+-------+
```

A CPU accesses a value in memory efficiently if the memory address of the value is an exact multiple of the size of the value being accessed. For example, for efficient access an 8-byte value is stored at a memory address which is a multiple of 8, and a 4-byte value is stored at a memory address which is a multiple of 4. A C++ compiler will normally place values in memory to respect this principle.

In C++, the declaration of an object (e.g. the `Person` entity) specifies the order in which data members are laid out in the object's block of memory. If the specified order would cause a data member to be incorrectly aligned for its size, the C++ compiler will insert padding bytes into the object to

enforce correct alignment. Such padding can make the object larger.

The option `entity_member_packing` instructs the OpenM++ compiler to order entity members from larger to smaller to minimize padding and reduce the size of the entity. If two members have the same alignment requirements, they are ordered lexicographically within that alignment requirement group.

In the RiskPaths model, turning entity member packing on changes the summary section of the above table to:

```
+-----------------------------------+-------+
| Sum of member bytes               |   699 |
| Bytes per entity                  |   728 |
| Storage efficiency (%)            |  96.0 |
+-----------------------------------+-------+
```

In RiskPaths, the size of Person decreased from 776 to 728 with entity member packing turned on.

By default, the OpenM++ compiler orders entity members to facilitate debugging in an IDE. The default order is

```
entity_id
time
age
all attributes declared in model code, ordered lexicographically
all other members, ordered lexicographically
```

If `entity_member_packing` is `on`, this default order is replaced by a non-intuitive order. So, for debugging sessions, it may be worthwhile to have entity member packing turned off. Note that by default `entity_member_packing` is `off`.

[back to topic contents]

Syntax and Use

By default, entity member packing is off. To activate it, include the statement

```
options entity_member_packing = on;
```

in the source code of a model. A typical place to insert this statement is the module `ompp_framework.ompp`.

[back to topic contents]

Entity Tables

This topic describes entity tables in depth. This topic is under construction and consists mostly of stub subtopics.

Related topics

- Model Code

Topic contents

- Introduction and concepts
- Table properties
- Snapshot tables
- Duration tables
- Run results
- Increments
- Accumulators
- Shorthand
- Increment Validity Increment validity
- Operators

Introduction and concepts

Content to follow. Topic outline is incomplete. The only complete subtopic is Increment Validity.

[back to topic contents]

Table properties

Table properties, ordered arbitrarily, follow the `table` keyword and precede the kind of entity which the table tabulates.

Possible values are

- `untransformed`
  Indicates that the table is to be computed ignoring weights and population scaling.
- `snapshot`
  Indicates that the table uses a filter which is `true` only instantaneously and that increments are values observed at those instantaneous moments. See section snapshot tables below for more.
- `screened[1-4]`
  Indicates that, after the simulation of a sub completes, table values are to be transformed using a model developer-supplied transformation function. See Screened Tables for more.

[back to topic contents]

Snapshot tables

under construction.

[back to topic contents]

Duration tables

under construction.

[back to topic contents]

Run results

Entity tables are computed for each sub, and run-level results computed from the sub-level results. Several methods are available to compute run-level results through the `measures_method` option, as follows:

`measures_method`	Description of run result
`average`	Measures are averaged across subs. ex #1: measure `count` run: `AVG(count)` ex #2: measure `income` run: `AVG(income)` ex #3: measure: `income/count` run: `AVG(income/count)`
`aggregate`	accumulators are averaged over subs, then the measure calculated using those. ex #1: measure: `count` run: `AVG(count)` ex #2: measure: `income` run: `AVG(income)` ex #3: measure: `income/count` run: `AVG(income)/(AVG(count)`
`assemble`	accumulators are summed over subs, and the measure computed using those ex #1: measure: `count` run: `SUM(count)` ex #2: measure: `income` run: `SUM(income)` ex #2: measure: `income/count` run: `SUM(income)/SUM(count)`

In these examples, `income` and `count` are sums which vary over subs, and the run result shows the operations performed on those subs to produce the run result.

[back to topic contents]

Increments

An increment is based on the value of an attribute when an entity enters a table cell and the value when it exits the cell. An entity enters a table cell at simulation entry or when the table filter becomes `true`. An entity leaves a table cell at simulation exit or when the table filter becomes `false`. An entity changes cell when attributes used in the classification dimensions of the table change value.

Keyword	Description
value_in	The attribute value when the entity enters the table cell.
value_out	The attribute value when the entity leaves the table cell.
delta	The difference in the attribute value between cell exit and cell entrance.
nz_value_in	The non-zero count of attribute value on entrance.
nz_value_out	The non-zero count of attribute value on exit.
nz_delta	The non-zero count of the difference in value between exit and entrance.
value_in2	The square of the value on entrance.
value_out2	The square of the value on exit.
delta2	The square of the difference in value between exit and entrance.

[back to topic contents]

Accumulators

Also referred to in this wiki as statistics. A table is weighted if the model was built with `weighted_tabulation` enabled and the table is not `untransformed`.

Keyword	Description	Notes
unit		Count of increments, weighted if the table is weighted
sum		Sum of increments, weighted if the table is weighted
minimum		

Keyword	Description	Notes
maximum		
mean		If unweighted, a numerically stable mean is computed. If weighted, a 'naive' estimate (quotient) is computed.
variance		Unweighted tables only. A numerically stable variance is computed.
stdev		Unweighted tables only. Derived from a numerically stable computed variance.
P2		Unweighted tables only.
P5		"
P10		"
P20		"
P25	Q1	"
P30		"
P40		"
P50	Median	"
P60		"
P70		"
P75	Q3	"
P80		"
P90		"
P95		"
P98		"
P99		"
gini		"

[back to topic contents]

Shorthand

These keywords can be a more compact way of specifying commonly-used combinations of increments and accumulators.

Keyword	Equivalent
delta(x)	sum(delta(x))
delta2(x)	sum(delta2(x))
nz_delta(x)	sum(nz_delta(x))
value_in(x)	sum(value_in(x))
value_in2(x)	sum(value_in2(x))
nz_value_in(x)	sum(nz_value_in(x))
value_out(x)	sum(value_out(x))
value_out2(x)	sum(value_out2(x))
nz_value_out(x)	sum(nz_value_out(x))
max_delta(x)	maximum(delta(x))

Keyword	Equivalent
max_value_in(x)	maximum(value_in(x))
max_value_out(x)	maximum(value_out(x))
min_delta(x)	minimum(delta(x))
min_value_in(x)	minimum(value_in(x))
min_value_out(x)	minimum(value_out(x))

[back to topic contents]

Increment Validity

This subtopic contains the following sections:

- Non-numeric floating point values
- Non-numeric values in OpenM++
- Increments and accumulators
- Invalid table increments
- Disabling table increment errors

[back to topic contents]

Non-numeric floating point values

A floating point number in OpenM++ is a C++ IEEE `double` or `float`, which is supported on most CPU hardware. It can hold an exact value like `123` or `0.5` or an approximation to a Real number like `0.1` or `pi`.

A floating point number can also hold one of three special non-numeric values: `+inf`, `-inf`, or `NaN` (indeterminate).

Non-numeric values arise naturally from arithmetic operations or function calls, e.g.

Expression	Result
`1.0 / 0.0`	`+inf`
`-1.0 / 0.0`	`-inf`
`0.0 / 0.0`	`NaN`
`log(0.0)`	`-inf`
`log(-1.0)`	`NaN`
`sqrt(-1.0)`	`NaN`
`exp(710.0)`	`+inf`

Note: The C++ specification for `std::exp` guarantees a result of +inf if the argument is greater than `709.8`.

Numeric values can sometimes result from floating point operations or function calls with non-numeric arguments, e.g.

Expression	Result
`1.0 / +inf`	`0.0`
`atan(+inf)`	`pi/2`
`exp(-inf)`	`0.0`
`exp(log(0.0))`	`0.0`

Logical comparison operators can have non-numeric arguments, e.g.

Expression	Result
`+inf > 42.0`	`true`
`-inf < +inf`	`true`
`NaN` < `+inf`	`false`
`NaN` == `NaN`	`false`
`NaN` != `NaN`	`false`

Note: The C++ specification states that all operators with NaN return NaN, including *all* comparison operators, notably the `==` operator in the preceding table. The C++ library contains functions to determine non-numeric values: `std::isnan` determines if a floating point number is NaN, `std::isinf` if it is +inf or -inf, and `std::isfinite()` if it is finite, i.e. a garden variety floating point value.

Arithmetic operations involving +inf or -inf typically result in non-numeric values, e.g.

Expression	Result
`+inf + 1.0`	`+inf`
`+inf - 1.0`	`+inf`
`+inf + +inf`	`+inf`
`sqrt(+inf)`	`+inf`
`inf / inf`	`NaN`

This is notably the case for `NaN`, which propagates in arithmetic operations and mathematical functions, e.g.

Expression	Result
`NaN + 1.0`	`NaN`
`+inf + NaN`	`NaN`
`sqrt(NaN)`	`NaN`

Sometimes operations involving `+inf` or `-inf` can produce `NaN`, e.g.

Expression	Result
`+inf - +inf`	`NaN`
`+inf / +inf`	`NaN`
`+inf / 0.0`	`NaN`

A general rule of thumb is that non-numeric floating point values propagate to results in arithmetic operations and mathematical functions.

[back to increment validity]
[back to topic contents]

Non-numeric values in OpenM++

Non-numeric values are used in several ways in OpenM++:

1. Global time is initialized to `-inf` before each case or replicate/sub.
2. All event times are initialized to `+inf` before each case or replicate/sub.
3. Derived parameters of floating point type are initialized to `NaN` (all cells).
4. An attribute of floating point type (`float`, `double`, `real`, or `Time`) can have a non-numeric value, depending on model logic.
5. A `maximum` table accumulator is initialized to `-inf`, and a `minimum` table accumulator to `+inf`.

[back to increment validity]
[back to topic contents]

Increments and accumulators

Each cell of an entity table contains one or more accumulators specified in the expression dimension of the table. As attributes change value during a run, increments are *pushed* to accumulators of the current cell of the table. An accumulator might be a running count of increments, the running sum of increments, the current maximum value, or a collection of all pushed values. When a run completes, statistics are *extracted* from the table accumulators for each cell to compute final values for that table cell. For example, the median `P50` for an attribute in a table cell is extracted from the accumulator underlying `P50`, which is a collection of all increments pushed to the cell during the run. The `P50` statistic is computed by sorting that collection and finding the middle value (or the average of the two middle values if the number of increments in the collection is even).

[back to increment validity]
[back to topic contents]

Invalid table increments

Some accumulators handle increments of +inf or -inf in an expected and natural way, e.g. an increment of +inf to a `maximum` accumulator, or an increment of +inf or -inf to a `P50` accumulator.

However, a non-numeric increment can cause an accumulator to become pegged to a non-numeric value. This is particularly true for a NaN increment.

Specifically,

1. Pushing an increment with value `+inf` or `-inf` to a `sum` or `gini` accumulator nullifies the effect of any previous or subsequent increment.
2. Pushing an increment with value `NaN` to *any* accumulator nullifies the effect of any previous or subsequent increments.

In other words, *an increment of one single entity, perhaps the result of a rarely occurring corner condition in model code, can cause an entire table cell to become empty*. OpenM++ treats that as an error in table design or model logic, halts the run, and writes a log message like

`Simulation error: Invalid increment -inf in table 'IncrementTestTable' using attribute 'my_dbl' on or after event 'MortalityEvent' in entity_id 208 in simulation member 0 with combined seed 1637697257 when current time is 88.06346668067070`

This particular error message manifested in a version of the `Alpha2` model which was modified to deliberately produce an increment error in the table

```
table Person IncrementTestTable
[integer_age >= 50]
{
   {
      value_out(my_dbl)
   }
};
```

The root cause of an invalid increment often occurs in a different event than the one responsible for pushing the increment. In the error message above, the invalid increment was detected when the entity was exiting the simulation after the `MortalityEvent`. The attribute `my_dbl` likely assumed a non-numeric value earlier in the simulation causing the invalid increment later.

The root cause of an invalid increment can be probed using Event Trace to examine the evolution of the specific attribute in the specific entity given in the runtime error message.

To enable event trace in the model, the following statement must be added to model code:

```
options event_trace = on;
```

and the model executable must be invoked with the argument `-OpenM.IniAnyKey`.

The following `EventTrace` settings (in an `ini` file) output all events and all changes in `my_dbl` in entity 208:

```
[OpenM]
LogToFile = true
TraceToFile = true

[EventTrace]
; format
ReportStyle = readable
NameColumnWidth = 20
; filters
SelectedEntities = 208
; events
ShowEvents = yes
; attributes
ShowAttributes = yes
SelectedAttributes = my_dbl
```

This produces the following trace file output:

```
      Time  Entity       Age      Id Trace            Value Name                  Remarks
  0.000000  Person  0.000000     208 ENTER
  0.000000  Person  0.000000     208   attr              0  my_dbl                initial
  0.000000  Person  0.000000     208  EVENT                 SpawnEvent
  0.500000  Person  0.500000     208  EVENT                 EyeColourChangeEvent
  1.000000  Person  1.000000     208  EVENT                 FirstBirthdayEvent
  2.632435  Person  2.632435     208  EVENT                 HappinessReversalEvent
  2.632435  Person  2.632435     208   attr           -inf  my_dbl                was 0
 10.105425  Person 10.105425     208  EVENT                 MoveEvent
 25.755745  Person 25.755745     208  EVENT                 HappinessReversalEvent
 26.741666  Person 26.741666     208  EVENT                 StartPlayingEvent
 30.141256  Person 30.141256     208  EVENT                 MoveEvent
 32.641206  Person 32.641206     208  EVENT                 MoveEvent
 34.927079  Person 34.927079     208  EVENT                 HappinessReversalEvent
 54.778775  Person 54.778775     208  EVENT                 HappinessReversalEvent
 59.496134  Person 59.496134     208  EVENT                 StartPlayingEvent
 60.500447  Person 60.500447     208  EVENT                 HappinessReversalEvent
 68.935493  Person 68.935493     208  EVENT                 MoveEvent
 76.110163  Person 76.110163     208  EVENT                 HappinessReversalEvent
 78.448787  Person 78.448787     208  EVENT                 StartPlayingEvent
 79.197382  Person 79.197382     208  EVENT                 MoveEvent
 87.282644  Person 87.282644     208  EVENT                 HappinessReversalEvent
 88.063467  Person 88.063467     208  EVENT                 MortalityEvent
 88.063467  Person 88.063467     208 EXIT
```

This trace output shows that the `my_dbl` attribute first assumed the non-numeric value `-inf` during the `HappinessReversalEvent` at an early age, before the entity was in scope of the table filter.

Here's the model code responsible for the invalid increment error later in the simulation:

```
entity Person
{
   double my_dbl;
   void update_funny_numbers(void);
   hook update_funny_numbers, HappinessReversalEvent, 43;
};

void Person::update_funny_numbers(void)
{
   double x = 0.0;
   my_dbl = std::log(x); // is -inf
}
```

If necessary, a Debug version of the model can be built and run with a conditional break point added to the first line of code in the event identified by Event Trace. In this example, the break point could be set to the first line of `HappinessReversalEvent`, with condition `entity_id == 208`.

Here's the model source code of the event which was the root cause of the invalid increment.

```
void Person::HappinessReversalEvent()
{
    happy = !happy;
    if ( !happy && playing ) {
        // stop playing if unhappy
        playing = FALSE;
    }
    if (happy && my_first_happy_time == TIME_INFINITE) {
        my_first_happy_time = time;
    }
    hook_HappinessReversalEvent();
}
```

When the break point is hit, execution can be stepped line by line in the debugger until the code responsible for setting `my_dbl` to a non-numeric value is found.

[back to increment validity]
[back to topic contents]

Disabling table increment errors

Normal handling of an invalid table increment can be controlled by the following option:

```
options verify_valid_table_increment = off;  // default is on
```

If this option is `off`, a warning like the following will be written to the log on each run:

`Warning : invalid table increment is not detected with verify_valid_table_increment = off`

A table with a non-numeric cell will display as empty in the UI and in `csv` export.

[back to increment validity]
[back to topic contents]

Operators

Keyword	Description
interval	
event	

[back to topic contents]

Enumerations

This topic is under construction and/or revision.

Topic summary, two sentences max.

Related topics

- Model Code

Topic contents

- Introduction and outline
- Enumerations as dimensions
- Enumeration index validity

Introduction and outline

Content to follow.

[back to topic contents]

Enumerations as dimensions

Each dimension of a multi-dimensional parameter, derived table or entity array is declared using an enumeration. In `RiskPaths`, for example,

```
parameters
{
   //EN Age baseline for first pregnancy
   double  AgeBaselinePreg1[AGEINT_STATE];
...
};
```

declares the 1-dimensional array parameter `AgeBaselinePreg1` using the enumeration `AGEINT_STATE`, which is a partition whose declaration is

```
partition AGEINT_STATE //EN 2.5 year age intervals
{
   15, 17.5, 20, 22.5, 25, 27.5, 30, 32.5, 35, 37.5, 40
};
```

In C++ model code, an element of an array can be referenced using an index. For example, in the following code fragment

```
TIME Person::timeFirstPregEvent()
{
...
   dHazard = AgeBaselinePreg1[age_status]
...
```

`AgeBaselinePreg1` is indexed by the attribute `age_status` to obtain an age-specific hazard.

Enumeration index validity

Index errors are a common source of errors in C++ programs. Such errors are typically not detected by the C++ compiler and often manifest only indirectly when the program is executed. They can be hard to detect and resolve, because they manifest outside of a program's logic, by reading or modifying unrelated locations in memory.

In model code,
an index is invalid if it falls outside the lower and upper integer limits of an enumeration dimension. For example, the parameter `ProbMort`

```
parameters
{
...
   double ProbMort[LIFE]; //EN Death probabilities
};
```

is declared using the range `LIFE`:

```
range LIFE //EN Simulated age range
{
    0,100
};
```

The model code fragment

```
TIME Person::timeDeathEvent()
{
...
    if (ProbMort[integer_age] >= 1)
...
}
```

would be invalid if `integer_age` was less than `0` or greater than `100`.

The option

```
options index_errors = on;
```

causes the OpenM++ compiler to generate code which verifies that all indices are valid during a model run.

Events

This topic describes events. This topic is under construction and consists mostly of stub subtopics.

Related topics

- Model Code
- Event Trace: Probe a model run at the micro level
- Time-like and Event-like Attributes: Definition of time-like and event-like attributes, and restrictions on use
- Censor Event Time: How to activate and use the censor_event_time optimization option
- Entity Function Hooks: Entity function hooks

Topic contents

under construction

- Introduction and concepts
- Event time function

Introduction and concepts

under construction

Declaration syntax

Event life-cycle.

Event implement function

under construction

Changes attributes when the event occurs.

Event time function

under construction

This subtopic contains the following sections:

- Return value
- Attributes affecting event time

[back to topic contents]

Return value

under construction

The return value of an event time function is the (conditional) time when the event will occur. It can be current time or future time. It can be infinity. If it is a time in the past of the entity a run-time error will occur.

The `WAIT` function. `WAIT(0)` means now. But other events may occur first, depending on event priority and event tie rules.

Calling a time function must not influence the state of the simulation, because the associated event has not occurred (yet). Model code which attempts to change an attribute during an event time calculation will cause a run-time error.

No side-effect means simulation framework can call freely.

Clock-like events and hazard-like events.

The event time is recalculated when specific attributes change.

An attempt to use a time-like attribute in an event time function causes a build-time error.

[back to event time function]
[back to topic contents]

Attributes affecting event time

When an attribute changes value, any event whose time depends on that attribute must have its occurrence time recalculated in order to remain valid.

To determine which attributes affect which events, the OpenM++ compiler scans the C++ model code in the body of event time functions for attribute names. The scan is not based on the logic of the code in the event time function, only on the presence of names of attributes. The names can be attributes of the entity or attributes of another entity referenced directly through a link.

Modgen specific: Modgen does not support event dependency on linked attributes and forbids links to attributes in event time functions.

Consider the following code fragment (adapted from the `Alpha2` test model):

```
entity Person
{
   //EN Integer age
   int integer_age = self_scheduling_int(age);
};
entity Thing
{
   //EN Count of celebratory birthday twirls performed
   int twirls = { 0 };
   //EN Do a twirl for the Person who spawned this Thing
   event timeTwirlEvent, TwirlEvent;
};

link Thing.spawner Person.things[];

TIME Thing::timeTwirlEvent()
{
   TIME event_time = TIME_INFINITE;
   if (spawner && (twirls < spawner->integer_age)) {
      event_time = WAIT(0); // twirl now!
   }
   return event_time;
}

void Thing::TwirlEvent()
{
   twirls++;
}
```

The `TwirlEvent` causes a `Thing` entity to twirl once on each birthday of the `Person` which spawned it.

The associated time function `timeTwirlEvent` uses three attributes:

- `spawner`, a link attribute of `Thing` which connects it to the `Person` entity which spawned it,
- `twirls`, an attribute of `Thing` which counts the number of times the `Thing` has twirled, and
- `spawner->integer_age`, a self-scheduling attribute of `Person` which increases by 1 on each birthday.

The OpenM++ compiler notes the use of these three attributes in the event time function and generates run-time code which calls `timeTwirlEvent` in a `Thing` entity if any of those three attributes changes value.

Specifically, when `integer_age` of a `Person` is incremented on a birthday, the event time of `TwirlEvent` of all `Thing` entities spawned by that `Person` are recalculated. The code in `timeTwirlEvent` causes the `TwirlEvent` to be scheduled immediately by returning `WAIT(0)`. After `TwirlEvent` is implemented and the twirl performed, `timeTwirlEvent` is called to schedule the next occurrence and returns infinity.

The OpenM++ compiler creates an output file which lists all attribute event dependencies. It is named `EventDependencies.csv` and is located in the `src` output folder, which in Windows is `MODEL/ompp/src` and in Linux is `MODEL/ompp-linux/src`.

For the `Alpha2` test model, `EventDependencies.csv` looks like this:

entity	event	attribute
Person	BlowHornEvent	blow_horns_now

entity	event	attribute
Person	EyeColourChangeEvent	eye_colour_definitive
Person	FirstBirthdayEvent	over_1
Person	MoveEvent	city
Person	SpawnEvent	spawning_done
Person	StartPlayingEvent	happy
Thing	BeingGoodEvent	making_trouble
Thing	TwirlEvent	spawner
Thing	TwirlEvent	twirls
Thing	TwirlEvent	spawner->integer_age
Thing	TwirlSpecialEvent	my_person1
Thing	TwirlSpecialEvent	twirls_special
Thing	TwirlSpecialEvent	my_person1->integer_age
Toy	DiscardEvent	lifetime

[back to event time function]
[back to topic contents]

Event scheduling

under construction

Extract from `OM_ROOT/include/omc/Event.h`:

```
/**
* event comparison. This is a true observer function but is not declared as const due to issues
* with get_event_id().
*
* @param [in,out] rhs The right hand side.
*
* @return true if the right-hand side is less than the left-hand side (this object).
*/
bool operator< ( BaseEvent& rhs )
{
   // earlier event time wins
   if ( event_time < rhs.event_time ) return true;
   if ( event_time > rhs.event_time ) return false;

   // higher event priority wins
   int event_priority = get_event_priority();
   int rhs_event_priority = rhs.get_event_priority();
   if ( event_priority > rhs_event_priority ) return true;
   if ( event_priority < rhs_event_priority ) return false;

   // lower event_id wins (earlier in alphabetic order by event name)
   int event_id = get_event_id();
   int rhs_event_id = rhs.get_event_id();
   if ( event_id < rhs_event_id ) return true;
   if ( event_id > rhs_event_id ) return false;

   // lower entity_id wins (created earlier)
   int entity_id = get_entity_id();
   int rhs_entity_id = rhs.get_entity_id();
   if ( entity_id < rhs_entity_id ) return true;
   else return false;
}
```

Self-scheduling events

under construction

Each entity has a built-in event which maintains all self-scheduling attributes in the entity.

Hooking to a self-scheduling attribute.

Tied events

under construction

The flow of time

under construction

Event loops

under construction

[back to topic contents]

Event Trace

This topic describes a model developer feature which reports the evolution of individual entities during a simulation.

Related topics

- Model Code
- Test Models

Topic contents

- Introduction and outline
- Quick start How to build and run a model with event trace capability
- Worked example 1 Using filters
- Worked example 2 Tracing attributes
- Worked example 3 Tracing links and multilinks
- Worked example 4 Tracing table increments
- General information General information
- Event trace columns The seven columns of event trace output
- Event trace messages The different kinds of messages in event trace output
- Event trace options Reference to all event trace options
- Trace file options Technical: Run options to specify trace file name, etc.
- Trace file API Technical: How to toggle trace output from model code

Introduction and outline

Event trace allows a model developer to probe the simulation of individual entities or groups of entities in detail. This can help to understand a model, to verify that it is working as intended, or to probe anomalies.

After an entity enters the simulation it undergoes a series of events which change attributes, either its own or those of linked entities. Changes in attributes can in turn condition the time of future events. An entity in a model with event trace capability generates event trace messages when it enters the simulation and during events. Event trace messages include simulation entry, simulation exit, event occurrence, conditional times of future events, and changes in attributes, links, and multilinks. Event trace messages are blocked or passed depending on run-time event trace options. Unblocked messages are written to the run trace file in a readable columnar format designed for visual scanning. Alternatively, messages can be written in a csv format to facilitate use in external applications.

Quick Start shows how to build a model with event trace capability and how to enable that capability in a model run.

The quick start is followed by several worked examples with illustrative inputs and outputs using the `RiskPaths` and `Alpha2` models:
Worked example 1 which illustrates tracing events,
Worked example 2 which illustrates tracing attributes,
Worked example 3 which illustrates tracing links and multilinks. and Worked example 4 which illustrates tracing table increments.

Please note that the output of the quick start and worked examples shown in this topic may differ slightly due to OpenM++ version differences.

The worked examples are followed by general information about event trace.

This is followed by three subtopics with reference material:
Event trace columns which describes the meaning of the output columns,
Event trace messages which describes all possible event trace messages,
and Event trace options which decribes all options to control and filter event trace messages.

This topic concludes with two specialized technical subtopics related to event trace.

[back to topic contents]

Quick start

This quick start example uses the `RiskPaths` model in `OM_ROOT/models/RiskPaths` in the OpenM++ distribution. By default a model is not built with event trace functionality, and without it the examples in this topic will not work.

This subtopic contains the following sections.

- 1. Build model with event trace capability
- 2. Create model `ini` file with event trace options
- 3. Run model using event trace

[continue to worked example 1]
[back to topic contents]

1. Build model with event trace capability

Edit the model source code file `RiskPaths/code/TraceOptions.mpp` to change the `event_trace` option from `off` to `on`:

```
options event_trace = on;
```

Build the Release version of RiskPaths.
In Windows, the model executable will be `RiskPaths/ompp/bin/RiskPaths.exe`.
In Linux, the model executable will be `RiskPaths/ompp-linux/bin/RiskPaths`.

[back to quick start]
[back to topic contents]

2. Modify model `ini` file with event trace options

In the same folder as the RiskPaths executable there may already be a copy of the default model `ini` file `RiskPaths.ini`. If not create it using your IDE or a text editor such as Notepad.

Edit `RiskPaths.ini` to have the following content:

```
[OpenM]
TraceToFile = true

[Parameter]
SimulationCases = 5

[EventTrace]
ReportStyle = readable
```

The following Windows screenshot shows `RiskPaths.ini` being edited in Visual Studio in the `RiskPaths` project:

[back to quick start]
[back to topic contents]

3. Run model using event trace

Launch the model in its `bin` directory using the `ini` file created in the previous step, being sure to also specify the `OpenM.IniAnyKey` option.

```
RiskPaths -ini RiskPaths.ini -OpenM.IniAnyKey true
```

In Windows you can run the Release version of RiskPaths from inside Visual Studio as follows:

- `Solution Configurations` to `Release` and `Solution Platforms` to `x64`
- `Project Properties > Configuration Properties > Debugging > Command Arguments` to `-ini RiskPaths.ini -OpenM.IniAnyKey true`
- `Project Properties > Configuration Properties > Debugging > Working Directory` to `$(TargetDir)`
- To launch the model, do `Debug > Start without debugging` or Press `Ctrl-F5`.

After the model runs the trace file `RiskPaths.trace.txt` should be present in the model `bin` directory and look like this:

```
     Time  Entity      Age      Id Trace          Value Name                          Remarks
 0.000000   Person  0.000000       1 ENTER
24.260999   Person 24.260999       1  EVENT            Union1FormationEvent
26.537813   Person 26.537813       1  EVENT            FirstPregEvent
27.260999   Person 27.260999       1  EVENT            UnionPeriod2Event
100.000000  Person 100.000000      1  EVENT             DeathEvent
100.000000  Person 100.000000      1 EXIT
 0.000000   Person  0.000000       2 ENTER
22.052373   Person 22.052373       2  EVENT            Union1FormationEvent
24.678078   Person 24.678078       2  EVENT            FirstPregEvent
25.052373   Person 25.052373       2  EVENT            UnionPeriod2Event
100.000000  Person 100.000000      2  EVENT             DeathEvent
100.000000  Person 100.000000      2 EXIT
 0.000000   Person  0.000000       3 ENTER
17.050111   Person 17.050111       3  EVENT            Union1FormationEvent
20.024665   Person 20.024665       3  EVENT            FirstPregEvent
20.050111   Person 20.050111       3  EVENT            UnionPeriod2Event
100.000000  Person 100.000000      3  EVENT             DeathEvent
100.000000  Person 100.000000      3 EXIT
 0.000000   Person  0.000000       4 ENTER
17.410717   Person 17.410717       4  EVENT            FirstPregEvent
100.000000  Person 100.000000      4  EVENT             DeathEvent
100.000000  Person 100.000000      4 EXIT
 0.000000   Person  0.000000       5 ENTER
24.157739   Person 24.157739       5  EVENT            FirstPregEvent
100.000000  Person 100.000000      5  EVENT             DeathEvent
100.000000  Person 100.000000      5 EXIT
```

The following Windows screenshot shows `RiskPaths.trace.txt` open in Visual Studio in the `RiskPaths` project. The screenshot also shows the command window and log output for the run, and highlights the expected warning generated by a model with event trace capability.

[back to quick start]
[back to topic contents]

Worked example 1

This section continues the quick start example to explore using EventTrace filters to find and probe entities in the simulation.

This subtopic contains the following sections:

- Find entities with a specific event
- Report on specific entities
- Report detailed history for a given entity
- Probe a time window

[back to topic contents]

Find entities with a specific event

RiskPaths simulates how first and second unions affect first birth frequency. To probe entities which experience a second union, change `SelectedEvents` to only show the event `Union2FormationEvent`.

```
[OpenM]
TraceToFile = true

[Parameter]
SimulationCases = 5

[EventTrace]
; format
ReportStyle = readable
NameColumnWidth = 20
; events
ShowEnterSimulation = no
ShowExitSimulation = no
ShowEvents = yes
SelectedEvents = Union2FormationEvent
```

The trace file `RiskPaths.trace.txt` now looks like this:

That's right, it's empty! That's because the run had only 5 cases and in those 5 cases the `Union2FormationEvent` never occurred. Expand the run to 10,000 cases by changing `SimulationCases` and set `MaximumLines` to report only the first 10. The width of the name column has been shortened to 20 from the default of 40 for more compact output.

```
[OpenM]
TraceToFile = true

[Parameter]
SimulationCases = 10000

[EventTrace]
; format
ReportStyle = readable
NameColumnWidth = 20
MaximumLines = 10
; events
ShowEnterSimulation = no
ShowExitSimulation = no
ShowEvents = yes
SelectedEvents = Union2FormationEvent
```

The trace file looks like this:

```
      Time  Entity       Age      Id Trace          Value Name                Remarks
    85.748654  Person  85.748654      11  EVENT              Union2FormationEvent
    46.987741  Person  46.987741      13  EVENT              Union2FormationEvent
    24.942941  Person  24.942941      65  EVENT              Union2FormationEvent
    26.208783  Person  26.208783      94  EVENT              Union2FormationEvent
    67.733676  Person  67.733676     101  EVENT              Union2FormationEvent
    36.152105  Person  36.152105     135  EVENT              Union2FormationEvent
    22.466353  Person  22.466353     211  EVENT              Union2FormationEvent
    20.010964  Person  20.010964     222  EVENT              Union2FormationEvent
    60.116048  Person  60.116048     262  EVENT              Union2FormationEvent
Maximum lines exceeded, increase using EventTrace.MaximumLines
```

[back to worked example 1]
[back to topic contents]

Report on specific entities

In the previous section, the `entity_id` of the youngest of the 10 is 222 and the oldest is 266. To probe those two entities in more detail, set `MaximumLines` to 1000, set `SelectedEvents` to empty to report on all events, and set `SelectedEntities` to report only on those two entities. Also, set `SimulationCases` to 500 to increase run speed (RiskPaths has only one entity in each case, so this is sufficient to simulate the two cases we're after).

```
[OpenM]
TraceToFile = true

[Parameter]
SimulationCases = 500

[EventTrace]
; format
ReportStyle = readable
NameColumnWidth = 20
MaximumLines = 1000
; filters
SelectedEntities = 222,266
; events
ShowEnterSimulation = no
ShowExitSimulation = no
ShowEvents = yes
SelectedEvents =
```

Re-run RiskPaths. The trace file now looks like this:

```
      Time   Entity       Age      Id Trace          Value Name                Remarks
  18.198452   Person  18.198452     222   EVENT              Union1FormationEvent
  19.944885   Person  19.944885     222   EVENT              Union1DissolutionEvent
  20.010964   Person  20.010964     222   EVENT              Union2FormationEvent
  20.212818   Person  20.212818     222   EVENT              FirstPregEvent
  21.198452   Person  21.198452     222   EVENT              UnionPeriod2Event
 100.000000   Person 100.000000     222   EVENT               DeathEvent
  39.395331   Person  39.395331     266   EVENT              Union1FormationEvent
  42.395331   Person  42.395331     266   EVENT              UnionPeriod2Event
  65.159506   Person  65.159506     266   EVENT              Union1DissolutionEvent
  87.325967   Person  87.325967     266   EVENT              Union2FormationEvent
 100.000000   Person 100.000000     266   EVENT               DeathEvent
```

The evolution of `entity_id` 222 looks a bit odd. RiskPaths documents `UnionPeriod2Event` as a 3-year period after first union formation which can affect fertility:

```
/*NOTE(Person.UnionPeriod2Event, EN)
   Clock event which changes the union duration state union_status from
   US_FIRST_UNION_PERIOD1 to US_FIRST_UNION_PERIOD2. This event occurs
   after 3 years in 1st union. The clock is set at first union formation.
*/
```

The event trace shows that for `entity_id` 222 a second union occurred at time=20.010964 and `UnionPeriod2Event` occurred later at time=21.198452 (exactly 3 years after the `Union1FormationEvent` event). It would be incoherent for `union_status`, which affects fertility, to change 3 years after the start of a first union if the Person is already in a second union. It might be good to investigate the model logic to confirm that the `UnionPeriod2Event` event has no effect if a second union has already formed.

[back to worked example 1]
[back to topic contents]

Report detailed history for a given entity

Probe `entity_id` 222 in more detail by setting `SelectedEntities` to 222 and observing scheduled future event times by turning on `ShowQueuedEvents`. All model events except for `DeathEvent` are listed in `SelectedEvents` to remove `DeathEvent` messages from the report. Otherwise, the report would be cluttered with lines showing the recomputation of time of death because the mortality rate changes at each birthday.

```
[OpenM]
TraceToFile = true

[Parameter]
SimulationCases = 500

[EventTrace]
; format
ReportStyle = readable
NameColumnWidth = 25
MaximumLines = 10000
; filters
SelectedEntities = 222
; events
ShowEnterSimulation = no
ShowExitSimulation = no
ShowEvents = yes
ShowQueuedEvents = yes
SelectedEvents = \
   FirstPregEvent, \
   Union1FormationEvent, \
   Union1DissolutionEvent, \
   Union2FormationEvent, \
   Union2DissolutionEvent, \
   UnionPeriod2Event
```

The line continuation character `\` is used to split the list of event names into multiple lines for readability and ease of editing.

The resulting trace report looks like this:

```
      Time  Entity      Age      Id Trace          Value Name                    Remarks
 15.000000  Person 15.000000     222   queued     81.828627   FirstPregEvent          was inf
 15.000000  Person 15.000000     222   queued     58.463729   Union1FormationEvent    was inf
 17.500000  Person 17.500000     222   queued     18.987406   FirstPregEvent          was 81.828627
 17.500000  Person 17.500000     222   queued     18.198452   Union1FormationEvent    was 58.463729
 18.198452  Person 18.198452     222  EVENT                Union1FormationEvent
 18.198452  Person 18.198452     222   queued     20.173365   FirstPregEvent          was 18.987406
 18.198452  Person 18.198452     222   queued     53.059542   Union1DissolutionEvent  was inf
 18.198452  Person 18.198452     222   queued          inf    Union1FormationEvent    was 18.198452
 18.198452  Person 18.198452     222   queued     21.198452   UnionPeriod2Event       was inf
 19.198452  Person 19.198452     222   queued     19.944885   Union1DissolutionEvent  was 53.059542
 19.944885  Person 19.944885     222  EVENT                Union1DissolutionEvent
 19.944885  Person 19.944885     222   queued     43.387166   FirstPregEvent          was 20.173365
 19.944885  Person 19.944885     222   queued          inf    Union1DissolutionEvent  was 19.944885
 19.944885  Person 19.944885     222   queued     20.010964   Union2FormationEvent    was inf
 20.000000  Person 20.000000     222   queued     55.173852   FirstPregEvent          was 43.387166
 20.010964  Person 20.010964     222  EVENT                Union2FormationEvent
 20.010964  Person 20.010964     222   queued     20.212818   FirstPregEvent          was 55.173852
 20.010964  Person 20.010964     222   queued     59.958810   Union2DissolutionEvent  was inf
 20.010964  Person 20.010964     222   queued          inf    Union2FormationEvent    was 20.010964
 20.212818  Person 20.212818     222  EVENT                FirstPregEvent
 20.212818  Person 20.212818     222   queued          inf    FirstPregEvent          was 20.212818
 20.212818  Person 20.212818     222   queued          inf    Union2DissolutionEvent  was 59.958810
 21.198452  Person 21.198452     222  EVENT                UnionPeriod2Event
 21.198452  Person 21.198452     222   queued          inf    UnionPeriod2Event       was 21.198452
```

No events of interest occurred in `entity_id` 222 after `UnionPeriod2Event`. On the other hand, this entity does not probe what might happen if a first birth had not already occurred prior to `UnionPeriod2Event`.

In this trace report, we see recalculation of future events triggered by other events. Recalculations are triggered by changes in an entity's attributes, and that only happens at events. The report shows, however, a recomputation of the time of event `Union1DissolutionEvent` occurring at time=19.198452, exactly one year after `Union1FormationEvent` occurs, with no associated event. This must be due to a self-scheduling event which is not shown in the output.

[back to worked example 1]
[back to topic contents]

Probe a time window

To probe in detail what's happening in `entity_id` 222 at time 19.198452 (the time of the event not shown in the previous run), turn on `ShowSelfSchedulingEvents` and `ShowQueuedSelfSchedulingEvents`, and set up a time restriction window which brackets that time by setting `MinimumTime=19.1` and `MaximumTime=19.2`.

```
[OpenM]
TraceToFile = true

[Parameter]
SimulationCases = 500

[EventTrace]
; format
ReportStyle = readable
NameColumnWidth = 45
MaximumLines = 10000
; filters
SelectedEntities = 222
MinimumTime = 19.1
MaximumTime = 19.2
; events
ShowEnterSimulation = no
ShowExitSimulation = no
ShowEvents = yes
ShowSelfSchedulingEvents = yes
ShowQueuedEvents = yes
ShowQueuedSelfSchedulingEvents = yes
SelectedEvents = \
   FirstPregEvent, \
   Union1FormationEvent, \
   Union1DissolutionEvent, \
   Union2FormationEvent, \
   Union2DissolutionEvent, \
   UnionPeriod2Event
```

The trace report looks like this:

```
     Time  Entity      Age      Id Trace          Value Name                                Remarks
 19.198452  Person 19.198452     222  EVENT                 self_scheduling_split(active_spell_duration(in_union,true),UNION_DURATION)
 19.198452  Person 19.198452     222   queued     21.198452   self_scheduling_split(active_spell_duration(in_union,true),UNION_DURATION)
 19.198452  Person 19.198452     222   queued     19.944885   Union1DissolutionEvent                was 53.059542
```

The missing event is revealed to be the self-scheduling attribute `self_scheduling_split(active_spell_duration(in_union,true),UNION_DURATION)`. Perusing the RiskPaths code reveals that this self-scheduling attribute is assigned to the identity attribute `union_duration`, which is in turn used in the computation of the time of the `Union1DissolutionEvent` event.

[back to worked example 1]
[back to topic contents]

Worked example 2

This subtopic illustrates tracing attributes. It contains the following sections:

- Attributes with event context
- Attributes without event context
- Find entities using an attribute
- Using the `case_seed` attribute

[back to topic contents]

Attributes with event context

Worked example 1 suggested that the derived attribute `union_duration` played a role in explaining the recomputation of an event time. Trace that attribute as well as `in_union` by turning on attribute tracing with `ShowAttributes`, and specifying the attributes to be traced with `SelectedAttributes`.

```
[OpenM]
TraceToFile = true

[Parameter]
SimulationCases = 500

[EventTrace]
; format
ReportStyle = readable
NameColumnWidth = 20
MaximumLines = 10000
; filters
SelectedEntities = 222
; events
ShowEnterSimulation = yes
ShowExitSimulation = yes
ShowEvents = yes
; attributes
ShowAttributes = yes
SelectedAttributes = \
   in_union, \
   union_duration
```

The trace output looks like this:

```
      Time   Entity       Age       Id Trace            Value Name                  Remarks
  0.000000   Person   0.000000      222 ENTER
  0.000000   Person   0.000000      222    attr              0   in_union              initial
  0.000000   Person   0.000000      222    attr              0   union_duration        initial
 18.198452   Person  18.198452      222   EVENT                  Union1FormationEvent
 18.198452   Person  18.198452      222    attr              1   in_union              was 0
 19.198452   Person  19.198452      222    attr              1   union_duration        was 0
 19.944885   Person  19.944885      222   EVENT                  Union1DissolutionEvent
 19.944885   Person  19.944885      222    attr              0   in_union              was 1
 19.944885   Person  19.944885      222    attr              0   union_duration        was 1
 20.010964   Person  20.010964      222   EVENT                  Union2FormationEvent
 20.010964   Person  20.010964      222    attr              1   in_union              was 0
 20.212818   Person  20.212818      222   EVENT                  FirstPregEvent
 21.010964   Person  21.010964      222    attr              1   union_duration        was 0
 21.198452   Person  21.198452      222   EVENT                  UnionPeriod2Event
 23.010964   Person  23.010964      222    attr              2   union_duration        was 1
 25.010964   Person  25.010964      222    attr              3   union_duration        was 2
 29.010964   Person  29.010964      222    attr              4   union_duration        was 3
 33.010964   Person  33.010964      222    attr              5   union_duration        was 4
100.000000   Person 100.000000      222   EVENT                  DeathEvent
100.000000   Person 100.000000      222 EXIT
```

The output shows the the initial values of the attributes when the entity enters the simulation and changes in attributes as the simulation progresses. The attribute `in_union` changes in response to union formation events as expected. The attribute `union_duration` changes with no preceding event message because self-scheduling event messages are blocked with these `EventTrace` settings.

[back to worked example 2]
[back to topic contents]

Attributes without event context

Event messages can be be removed from the output by setting `ShowEvents` to `off`. The following settings trace some key attributes of `RiskPaths` fo a single case.

```
[OpenM]
TraceToFile = true

[Parameter]
SimulationCases = 500

[EventTrace]
; format
ReportStyle = readable
NameColumnWidth = 20
MaximumLines = 10000
; filters
SelectedEntities = 222
; events
ShowEnterSimulation = no
ShowExitSimulation = no
ShowEvents = no
; attributes
ShowAttributes = yes
SelectedAttributes = \
   case_seed, \
   union_status, \
   parity_status, \
   union_duration, \
   dissolution_duration
```

The trace output looks like this:

```
      Time  Entity      Age      Id Trace          Value Name                 Remarks
  0.000000  Person  0.000000     222  attr     1214330268  case_seed             initial
  0.000000  Person  0.000000     222  attr              0  dissolution_duration  initial
  0.000000  Person  0.000000     222  attr              0  parity_status         initial
  0.000000  Person  0.000000     222  attr              0  union_duration        initial
  0.000000  Person  0.000000     222  attr              0  union_status          initial
 18.198452  Person 18.198452     222  attr              1  union_status          was 0
 19.198452  Person 19.198452     222  attr              1  union_duration        was 0
 19.944885  Person 19.944885     222  attr              0  union_duration        was 1
 19.944885  Person 19.944885     222  attr              3  union_status          was 1
 20.010964  Person 20.010964     222  attr              4  union_status          was 3
 20.212818  Person 20.212818     222  attr              1  parity_status         was 0
 21.010964  Person 21.010964     222  attr              1  union_duration        was 0
 23.010964  Person 23.010964     222  attr              2  union_duration        was 1
 25.010964  Person 25.010964     222  attr              3  union_duration        was 2
 29.010964  Person 29.010964     222  attr              4  union_duration        was 3
 33.010964  Person 33.010964     222  attr              5  union_duration        was 4
```

[back to worked example 2]
[back to topic contents]

Find entities using an attribute

In this example, we use event trace to identify entities which were pregnant during a second union, and output a detailed history of the oldest.

First, we add the new attribute `my_filter` to `RiskPaths` by editing `Fertility.mpp` and adding the following lines:

```
actor Person
{
   //EN Pregnant in second union
   bool my_filter = (parity_status == PS_PREGNANT) && (union_status == US_SECOND_UNION);
};
```

and rebuild `RiskPaths`. Next we set event trace options to block all messages except attribute messages for `my_filter` which have the value `true`. In C++ `true` has the value 1 and `false` has the value 0.

```
[OpenM]
TraceToFile = true

[Parameter]
SimulationCases = 500

[EventTrace]
; format
ReportStyle = readable
NameColumnWidth = 20
MaximumLines = 10000
; filters
; events
ShowEnterSimulation = no
ShowExitSimulation = no
ShowEvents = no
; attributes
ShowAttributes = yes
SelectedAttributes = my_filter
MinimumAttribute = 1
MaximumAttribute = 1
```

The output is:

```
      Time   Entity       Age       Id Trace            Value Name                  Remarks
 26.196550   Person  26.196550       65    attr                1   my_filter            was 0
 26.395338   Person  26.395338       94    attr                1   my_filter            was 0
 24.133799   Person  24.133799      211    attr                1   my_filter            was 0
 20.212818   Person  20.212818      222    attr                1   my_filter            was 0
 33.073613   Person  33.073613      279    attr                1   my_filter            was 0
 32.492563   Person  32.492563      479    attr                1   my_filter            was 0
 23.148798   Person  23.148798      481    attr                1   my_filter            was 0
```

The output displays the 7 entities among the 500 which satisfied the condition during their lifetime. The `Id` column shows the `entity_id` of each, and the `Age` column shows the age at which the condition first become true.

Among the 7, the oldest was `entity_id` 279.

The following settings probe the lifetime of that entity in more detail:

```
[OpenM]
TraceToFile = true

[Parameter]
SimulationCases = 500

[EventTrace]
; format
ReportStyle = readable
NameColumnWidth = 20
MaximumLines = 10000
; filters
SelectedEntities = 279
; events
ShowEnterSimulation = yes
ShowExitSimulation = yes
ShowEvents = yes
```

The output is

```
      Time   Entity       Age       Id Trace            Value Name                  Remarks
  0.000000   Person  0.000000      279 ENTER                                 initial time=0.000000,age=0.000000
 27.167907   Person  27.167907      279   EVENT                Union1FormationEvent
 29.155224   Person  29.155224      279   EVENT                Union1DissolutionEvent
 30.167907   Person  30.167907      279   EVENT                UnionPeriod2Event
 32.105193   Person  32.105193      279   EVENT                Union2FormationEvent
 33.073613   Person  33.073613      279   EVENT                FirstPregEvent
100.000000   Person 100.000000      279   EVENT                DeathEvent
100.000000   Person 100.000000      279 EXIT
```

[back to worked example 2]
[back to topic contents]

Using the `case_seed` attribute

The example Attributes without event context above traced `case_seed`, which is a built-in attribute of a case-based model. Case seeds of interest

can also be obtained from aggregate output tables using an expression like `max_value_out(case_seed)` to identify the seed of a case for each populated cell of a table.

`case_seed` can be useful to speed exploration of a specific case in a case-based model by arranging that a run simulate only that single case. The following settings set the parameter `SimulationSeed` to 1214330268 from the previous trace output. The number of cases is also reduced from 500 to 1, for a very fast run. The `SelectedEntities` filter was removed.

```
[OpenM]
TraceToFile = true

[Parameter]
SimulationCases = 1
SimulationSeed = 1214330268

[EventTrace]
; format
ReportStyle = readable
NameColumnWidth = 20
MaximumLines = 10000
; filters
; events
ShowEnterSimulation = no
ShowExitSimulation = no
ShowEvents = no
; attributes
ShowAttributes = yes
SelectedAttributes = \
   case_seed, \
   union_status, \
   parity_status, \
   union_duration, \
   dissolution_duration
```

With these settings, the trace file looks like this:

Time	Entity	Age	Id	Trace	Value	Name	Remarks
0.000000	Person	0.000000	1	attr	1214330268	case_seed	initial
0.000000	Person	0.000000	1	attr	0	dissolution_duration	initial
0.000000	Person	0.000000	1	attr	0	parity_status	initial
0.000000	Person	0.000000	1	attr	0	union_duration	initial
0.000000	Person	0.000000	1	attr	0	union_status	initial
18.198452	Person	18.198452	1	attr	1	union_status	was 0
19.198452	Person	19.198452	1	attr	1	union_duration	was 0
19.944885	Person	19.944885	1	attr	0	union_duration	was 1
19.944885	Person	19.944885	1	attr	3	union_status	was 1
20.010964	Person	20.010964	1	attr	4	union_status	was 3
20.212818	Person	20.212818	1	attr	1	parity_status	was 0
21.010964	Person	21.010964	1	attr	1	union_duration	was 0
23.010964	Person	23.010964	1	attr	2	union_duration	was 1
25.010964	Person	25.010964	1	attr	3	union_duration	was 2
29.010964	Person	29.010964	1	attr	4	union_duration	was 3
33.010964	Person	33.010964	1	attr	5	union_duration	was 4

The trace output (and simulation) is identical to the previous output except for the `Id` column, which changed from 222 to 1. The `Id` column shows the `entity_id` of the entity which produced the message. In a case-based model, `entity_id` is a unique sequentially increasing counter for entities in the run.

Case-based models which construct cases by reading a file of microdata may not be able to reproduce a case of interest using this technique. `OzProj` in the OpenM++ distribution is an example. Such models can filter on one or more case seeds directly using `SelectedCaseSeeds`, as in the following example.

```
[OpenM]
TraceToFile = true

[Parameter]
SimulationCases = 500

[EventTrace]
; format
ReportStyle = readable
NameColumnWidth = 20
MaximumLines = 10000
; filters
SelectedCaseSeeds = 1214330268
; events
ShowEnterSimulation = no
ShowExitSimulation = no
ShowEvents = no
; attributes
ShowAttributes = yes
SelectedAttributes = \
   case_seed, \
   union_status, \
   parity_status, \
   union_duration, \
   dissolution_duration
```

With these options, the output is as before.

```
      Time  Entity      Age      Id Trace          Value Name                Remarks
  0.000000  Person  0.000000     222  attr     1214330268  case_seed            initial
  0.000000  Person  0.000000     222  attr              0  dissolution_duration  initial
  0.000000  Person  0.000000     222  attr              0  parity_status        initial
  0.000000  Person  0.000000     222  attr              0  union_duration       initial
  0.000000  Person  0.000000     222  attr              0  union_status         initial
 18.198452  Person 18.198452     222  attr              1  union_status         was 0
 19.198452  Person 19.198452     222  attr              1  union_duration       was 0
 19.944885  Person 19.944885     222  attr              0  union_duration       was 1
 19.944885  Person 19.944885     222  attr              3  union_status         was 1
 20.010964  Person 20.010964     222  attr              4  union_status         was 3
 20.212818  Person 20.212818     222  attr              1  parity_status        was 0
 21.010964  Person 21.010964     222  attr              1  union_duration       was 0
 23.010964  Person 23.010964     222  attr              2  union_duration       was 1
 25.010964  Person 25.010964     222  attr              3  union_duration       was 2
 29.010964  Person 29.010964     222  attr              4  union_duration       was 3
 33.010964  Person 33.010964     222  attr              5  union_duration       was 4
```

If a case-based model is designed to generate multiple cloned entities within a case, `SelectedCaseSeeds` can be helpful to probe all clones within a case.

[back to worked example 2]
[back to topic contents]

Worked example 3

This subtopic illustrates tracing link attributes and multilinks. It contains the following sections:

- Links and multiple entity types
- Multilinks
- Expanding the selected entities

[back to topic contents]

Links and multiple entity types

This section uses the `Alpha2` model, which is part of the OpenM++ distribution. To make `Alpha2` capable of event trace, the statement

```
options event_trace = on;
```

was added to the model source module `ompp_framework.ompp`, and a default `ini` file `Alpha2.ini` added to the model `bin` folder, as described in Quick start. Also as described in Quick start, the model command line arguments were set to
`-ini Alpha2.ini -OpenM.IniAnyKey true`

`Alpha2` has several different kinds of entities, two of which are `Person` and `Thing`. A one-to-one link between a `Person` and a `Thing` is declared in the model source statement

```
link Person.my_thing1 Thing.my_person1;
```

which declares the link attribute `my_thing1` in `Person` and the reciprocal link `my_person1` in `Thing`.

Both `Person` and `Thing` have an event named `MortalityEvent`. `Person` has a `SpawnEvent` which occurs immediately after a `Person` enters the simulation and which creates other entities.

The following run settings in `Alpha2.ini` trace selected events and attributes for entities 1 and 2.

```
[OpenM]
TraceToFile = true

[Parameter]
SimulationCases = 1

[EventTrace]
; format
ReportStyle = readable
NameColumnWidth = 20
; filters
SelectedEntities = 1,2
; events
ShowEnterSimulation = yes
ShowExitSimulation = yes
ShowEvents = yes
SelectedEvents = \
   SpawnEvent, \
   MortalityEvent
; attributes
ShowAttributes = yes
SelectedAttributes = \
   my_thing1, \
   my_person1
```

The resulting trace output looks like this:

```
The answer is  42!
        Time   Entity        Age       Id Trace              Value Name                    Remarks
    0.000000   Person   0.000000        1 ENTER
    0.000000   Person   0.000000        1     link         nullptr   my_thing1             initial
    0.000000   Person   0.000000        1   EVENT                    SpawnEvent
    0.000000    Thing   0.000000        2 ENTER
    0.000000    Thing   0.000000        2     link         nullptr   my_person1            initial
    0.000000   Person   0.000000        1     link               2   my_thing1             was nullptr
    0.000000    Thing   0.000000        2     link               1   my_person1            was nullptr
   48.288955   Person  48.288955        1   EVENT                    MortalityEvent
   48.288955   Person  48.288955        1 EXIT
   48.288955    Thing  48.288955        2     link         nullptr   my_person1            was 1
  117.760965    Thing 117.760965        2   EVENT                    MortalityEvent
  117.760965    Thing 117.760965        2 EXIT
```

The first line in the trace output is produced by model code in `Alpha2` which tests the trace file API. It is irrelevant to this example, but does illustrate that the trace file can contain output from model code, not just from event trace.

The kind of entity (`Person` or `Thing`) is shown in the `Entity` column and the `entity_id` which produced the message is shown in the `Id` column. `MortalityEvent` is actually two different events with the same name, one in `Person` and one in `Thing`. `SpawnEvent` in `Person` 1 creates a `Thing` (`entity_id` 2) and links to it through the link attribute `my_thing1`. Here's the corresponding model code extract:

```
void Person::SpawnEvent()
{
   // Create things and add to simulation
   Thing *thing1 = new Thing;
   thing1->Start();

   // Link Person to thing1
   my_thing1 = thing1;
...
```

When `MortalityEvent` causes `Person` 1 to exit the simulation, all its links are automatically emptied. That causes the link `my_person1` in `Thing` 2 to become `nullptr` as shown in the trace output above.

[back to worked example 3]
[back to topic contents]

Multilinks

In `Alpha2` each `Person` can have more than one `Thing`. The relationship is declared in the one-to-many `link` statement in model code

```
link Thing.spawner Person.things[];
```

which associates the link attribute `spawner` in `Thing` to the `Person` that spawned the `Thing`. A `Person` has a reciprocal multilink `things` which contains all the `Thing` entities it spawned. The following model code extract adds two `Thing` entities to the `things` multilink:

```
void Person::SpawnEvent()
{
   // Create things and add to simulation
   Thing *thing1 = new Thing;
   thing1->Start();

   Thing *thing2 = new Thing;
   thing2->Start();

   // populate multi-link of things
   things->Add(thing1);
   things->Add(thing2);
...
}
```

The statement `things->Add(thing1);` automatically assigns the reciprocal `spawner` attribute in `Thing` to the `Person` entity.

The following run settings explore the multilink `things` and the reciprocal `spawner` link attribute.

```
[OpenM]
TraceToFile = true

[Parameter]
SimulationCases = 1

[EventTrace]
; format
ReportStyle = readable
NameColumnWidth = 20
; filters
SelectedEntities = 1,2,13
; events
ShowEnterSimulation = yes
ShowExitSimulation = yes
ShowEvents = yes
SelectedEvents = \
   SpawnEvent, \
   MortalityEvent
; attributes
ShowAttributes = yes
SelectedAttributes = \
   spawner, \
   things
```

The resulting trace output is:

```
The answer is  42!
        Time   Entity        Age       Id Trace              Value Name                  Remarks
    0.000000   Person   0.000000        1 ENTER
    0.000000   Person   0.000000        1     multi                 things                initial {}
    0.000000   Person   0.000000        1   EVENT                 SpawnEvent
    0.000000    Thing   0.000000        2 ENTER
    0.000000    Thing   0.000000        2     link         nullptr   spawner               initial
    0.000000    Thing   0.000000       13 ENTER
    0.000000    Thing   0.000000       13     link         nullptr   spawner               initial
    0.000000   Person   0.000000        1     multi++           2   things                is {2}
    0.000000    Thing   0.000000        2     link              1   spawner               was nullptr
    0.000000   Person   0.000000        1     multi++          13   things                is {2,13}
    0.000000    Thing   0.000000       13     link              1   spawner               was nullptr
   48.288955   Person  48.288955        1   EVENT                 MortalityEvent
   48.288955   Person  48.288955        1 EXIT
   48.288955    Thing  48.288955        2     link         nullptr   spawner               was 1
   48.288955    Thing  48.288955       13     link         nullptr   spawner               was 1
  117.760965    Thing 117.760965        2   EVENT                 MortalityEvent
  117.760965    Thing 117.760965        2 EXIT
  144.675208    Thing 144.675208       13   EVENT                 MortalityEvent
  144.675208    Thing 144.675208       13 EXIT
```

Each trace message for the `things` multilink shows the current contents of the multilink in the `Remarks` column. The initial value of `things` is empty, and `multi++` in the `Trace` column notes when an entity is added to the multilink. There are no `multi--` messages in this example, because the two `Thing` entities happened to outlive the `Person` entity. The `spawner` link attribute in each `Thing` was set automatically to `nullptr` when the `Person` exited the simulation during age 48.

Incidentally, the second `Thing` has `entity_id` 13 because each `Thing` spawns 10 `Toy` entities, but that's another story!

[back to worked example 3]
[back to topic contents]

Expanding the selected entities

This example uses the option `SelectLinkedEntities` to automatically select entities added through a link or multilink to an active selected entity. The following settings set selected entities to the single entity with 'entity_id' 1.

```
[OpenM]
TraceToFile = true
ProgressPercent = 25

[Parameter]
SimulationCases = 1

[EventTrace]
; format
ReportStyle = readable
NameColumnWidth = 20
; filters
SelectedEntities = 1
SelectLinkedEntities = yes
; events
ShowEnterSimulation = yes
ShowExitSimulation = yes
ShowEvents = yes
SelectedEvents = \
   SpawnEvent, \
   MortalityEvent
```

The trace output is

```
The answer is  42!
        Time   Entity        Age       Id Trace              Value Name                  Remarks
    0.000000   Person   0.000000        1 ENTER
    0.000000   Person   0.000000        1   EVENT                 SpawnEvent
    0.000000   Person   0.000000        1  selected++           2   my_thing1
    0.000000   Person   0.000000        1  selected++          13   my_thing2
   48.288955   Person  48.288955        1   EVENT                 MortalityEvent
   48.288955   Person  48.288955        1 EXIT
  117.760965    Thing 117.760965        2   EVENT                 MortalityEvent
  117.760965    Thing 117.760965        2 EXIT
  144.675208    Thing 144.675208       13   EVENT                 MortalityEvent
  144.675208    Thing 144.675208       13 EXIT
```

The messages with `selected++` in the `Trace` column show two entities 2 and 13 added to the set of selected entities. Subsequent messages include contributions from the three entities 1,2, and 13. Had entity 2 or 3 added links to other entities during the simulation, the set of selected

entities would have been expanded to include them, too.

[back to worked example 3]
[back to topic contents]

Worked example 4

This worked example illustrates Event Trace options which show `RiskPaths` pushing table increments to accumulators for the entity table `T06_BirthsByUnion`. Generally, model developers don't need to probe the low-level mechanics of entity tabulation in OpenM++, but it can sometimes be useful.
Here's the declaration of that table:

```
table Person T06_BirthsByUnion  //EN Pregnancies by union status & order
[trigger_entrances(parity_status, PS_PREGNANT)]
{
  {
    unit    //EN Number of pregnancies
  }
  * union_status+  //EN Union Status at pregnancy
};
```

With a 5,000 case run, that table looks like this:

Union Status at pregnancy	Number of pregnancies
Never in union	1,285
First union < 3 years	2,986
First Union > 3 years	293
After first union	11
Second union	57
After second union	1
All	4,633

When analyzing table increments, it's generally best to restrict the table to a single entity of interest, because otherwise the output may be too voluminous to be useful.

Here's what that table looks like in a run with only 1 case:

Union Status at pregnancy	Number of pregnancies
Never in union	0
First union < 3 years	1
First Union > 3 years	0
After first union	0
Second union	0
After second union	0
All	1

The table shows that the one `Person` in the run had a single pregnancy which occurred in the first 3 years of the first union. The marginal total over all `union_status` categories is necessarily also 1.

The following run settings trace events, increments to this table, and changes to the two attributes `parity_status` and `union_status` used in the table declaration:

```
[OpenM]
TraceToFile = true

[Parameter]
SimulationCases = 1

[EventTrace]
ReportStyle = readable
NameColumnWidth = 25
MaximumLines = 20000
ShowEnterSimulation = yes
ShowExitSimulation = yes
ShowAttributes = yes
SelectedAttributes = parity_status, union_status
ShowEvents = yes
ShowTableIncrements = yes
SelectedTables = T06_BirthsByUnion
```

With these settings, the trace output looks like this:

```
      Time   Entity       Age       Id Trace            Value Name                      Remarks
  0.000000   Person  0.000000        1 ENTER
  0.000000   Person  0.000000        1    attr            0   parity_status            initial
  0.000000   Person  0.000000        1    attr            0   union_status             initial
 24.260999   Person 24.260999        1   EVENT                Union1FormationEvent
 24.260999   Person 24.260999        1    attr            1   union_status             was 0
 26.537813   Person 26.537813        1   EVENT                FirstPregEvent
 26.537813   Person 26.537813        1    attr            1   parity_status            was 0
 27.260999   Person 27.260999        1   EVENT                UnionPeriod2Event
 27.260999   Person 27.260999        1   INCREMENT           1   T06_BirthsByUnion.acc0     cell=[1] accumulator=1
 27.260999   Person 27.260999        1   INCREMENT           1   T06_BirthsByUnion.acc0     cell=[6] accumulator=1
 27.260999   Person 27.260999        1    attr            2   union_status             was 1
100.000000   Person 100.000000       1   EVENT                DeathEvent
100.000000   Person 100.000000       1 EXIT
```

The trace output is coherent with the table shown above: The `Person` experienced a `FirstPregEvent` at age `26.537813`, when `union_status` was at its second level `First union < 3 years` (which has integer value 1).

The event trace shows two `INCREMENT` lines. The first pushes an increment with value 1 to cell `[1]` of the table, which is correct (index `[1]` is the second cell, corresponding to `First union < 3 years`). The second pushes the same increment to cell `[6]` of the table. That's correct because `[6]` is the margin index for the table dimension. An increment is always pushed to the body and to all margin dimensions, and to all crossings of margin dimensions. For example, if a table has two dimensions and both have a margin, an increment will be pushed to 4 cells: one for the table body, one for each margin, and one for the crossed margins.

The `INCREMENT` lines above are correct, but their timing in the `Person` timeline may be puzzling. That's because by design a table increment is pushed to an accumulator 'lazily'. The push happens only when an upcoming attribute change would invalidate the pending increment. Pushes can also happen when the entity exits the simulation. So, an increment may appear in the timeline later than might be expected. You can see that in action in the event trace output above when an upcoming change to `union_status` at time `27.260999` causes the pending increment to be pushed immediately before `union_status` changes. That increment was originally created when the table filter flashed `true` at time `26.537813` when the pregnancy occurred.

Table increments are 'lazy' to ensure that changes in multiple classificatory dimensions and/or filter settle down before being treated as a single finalized and coherent table increment.

For `INCREMENT` rows, the Value column is the value of the increment being pushed to an accumulator (always 1 in this example because 'unit'). There are two increments shown, one is for the body of the table, the other for the margin. Each cell of a table has an accumulator (more than one if table expressions use multiple accumulators). The cell indices of the increment/accumulator are shown in the Remarks column, as is the value of the accumulator after the increment (from all cases so far in the sub). The two accumulators shown in the trace output are both 1 because the run had no other cases.

[back to topic contents]

General information

A model built with event trace capability can be run repeatedly with different trace options with no need to rebuild.

Event trace works with Release versions of models, so can be used to probe details in large simulations.

Event trace is intended for model development, not production. A model built with event trace will output the following warning to the log whenever it is run:

```
Warning : possible performance impact - model built with event_trace = on
```

A model built with event trace will also output the following warning to the log whenever it is run:

```
Warning : model can expose microdata at run-time with event_trace = on
```

If this is not a concern, for example the model generates a synthetic population, this warning can be disabled by the following statement:

```
options event_trace_warning = off;
```

Some entity attributes are created by the OpenM++ compiler to implement model functionality. For example, if an entity table has a filter, an identity attribute is created to implement it. These internal generated attributes are normally hidden but they can be made visible by the following statement:

```
options all_attributes_visible = on;
```

An event trace message can only be produced by an active entity. Event trace messages are not produced before the entity enters the simulation or after the entity exits the simulation.

Event trace messages are produced directly and immediately as model code executes. The messages are output to the trace file if `EventTrace` filter conditions are met. For example, if an attribute is selected, an event trace message will be produced immediately whenever the valus changes. If the attribute is changed more than once during the implementation of an event, each change will produce a separate message.

Trace output is disabled by default when a model is run. Use `OpenM.TraceToFile` to enable it, e.g.

```
[OpenM]
TraceToFile = true
```

See the subtopic Trace file options for a complete listing of trace file options.

Event trace options are processed only if the model is run with the command line option `-OpenM.IniAnyKey true`. Unlike other options, `OpenM.IniAnyKey` must be specified on the command line, not in an `ini` file.

To avoid confusing output, event trace should be used in runs with a single sub/replicate/member.

A model can write lines to the trace file directly, in which case those lines will be interleaved with any event trace messages.

[back to topic contents]

Event trace columns

Each event trace message has up to 8 columns of information. Values for `Time`, `Entity`, `Age`, `Id`, and `Trace` are always present. Values for `Value`, `Name`, and `Remarks` may be absent, depending on the nature of the message.

The order, left- or right-justification, capitalization, and indentation of columns varies by message to help peruse voluminous output for salient features.

Column	Column header	Description
1	Time	The time of the entity when it produced the message.
2	Entity	The type of entity which produced the message, e.g. `Person`.
3	Age	The age of the entity when it produced the message.
4	Id	The `entity_id` of the entity which produced the message.
5	Trace	The kind of message, e.g. `EVENT` is a message for event occurrence. See Event trace messages for all possible values.
6	Value	A value associated with the message, e.g. the current value of an attribute.
7	Name	A name associated with the message, e.g. the name of the event or attribute.

Column	Column header	Description
8	Remarks	Supplementary information, e.g. the value of the attribute before it changed, or the contents of a multilink

If `ReportStyle` is `csv`, an additional leading column `Line` is present. For more information, see Event trace options - format.

[back to topic contents]

Event trace messages

The following table lists all possible kinds of event trace message. Values noted as 'initial' are those when the entity first enters the simulation before it experiences any events. The initial value of time does not necessarily correspond to the global time when the message was produced.

Kind of message	Trace (5)	Value (6)	Name (7)	Remarks (8)
Entity enters simulation	`ENTER`			
Entity exits simulation	`EXIT`			
Event occurrence	`EVENT`		event name	
Event future time	`queued`	The future time	event name	The previous value of the future event time.
Attribute - initial	`attr`	The initial value	attribute name	
Attribute - change	`attr`	The current value	attribute name	The previous value of the attribute.
Link - initial	`link`	The `entity_id` or `nullptr`	link name	
Link - change	`link`	The `entity_id` or `nullptr`	link name	The previous value of the link.
Multilink - initial	`multi`		multilink name	A list of all entities in the multilink.
Multilink - add	`multi++`	The `entity_id` of the added entity	multilink name	A list of all entities in the multilink.
Multilink - remove	`multi--`	The `entity_id` of the removed entity	multilink name	A list of all entities in the multilink.
Entity added to list of selected entities	`selected++`	The `entity_id` of the added entity	link or multilink name	
Table increment pushed to accumulator	`INCREMENT`	The value of the increment	table name and accumulator #	The table cell indices of the accumulator to which the increment was pushed, and the updated value of the accumulator.

[back to topic contents]

Event trace options

Event trace options fall into five broad categories which are described in the following sections.

- Format `ReportStyle`, `MaximumLines`, `NameColumnWidth`
- Filters Block/pass messages based on entity characteristics

- Events Block/pass messages on events and entity life cycle
- Attributes Block/pass messages on attribute changes
- Increments Block/pass messages on push table increment to accumulator

For reference, here is an extract of a model run `.ini` file with all event trace options:

```
;#==========================================================================
;#
;# event trace model development options
;#
;# Requires activation of model development options using -OpenM.IniAnyKey (see above).
;# Requires that model code contains the statement
;#   options event_trace = on;
;# Requires OpenM.TraceToFile = true
;#
;# See wiki for explanation of EventTrace options
;
; [EventTrace]
;
; format
;
; ReportStyle = readable            ; "modgen", "readable", or "csv", default: modgen
; MaximumLines = 100000             ; integer value, default: 20000
; NameColumnWidth = 20              ; integer value, default: 40
;
; filters
;
; SelectedEntityKinds = e1,e2,e3      ; comma separated list of entity kinds, if empty all entity kinds
; SelectedEntities = 1,2,3           ; comma separated list of integers, if empty all entities
; SelectLinkedEntities = no          ; default: no
; SelectedCaseSeeds = 1,2,3          ; comma separated list of case seeds, if empty all cases
; MinimumTime = 2025                 ; double value, default: -inf
; MaximumTime = 2025                 ; double value, default: +inf
; MinimumAge = 65                    ; double value, default: -inf
; MaximumAge = 66                    ; double value, default: +inf
;
; events
;
; ShowEnterSimulation = yes          ; default: yes
; ShowExitSimulation = yes           ; default: yes
; ShowEvents = yes                   ; default: yes
; SelectedEvents =  e1,e2,e3         ; comma separated list of event names, if empty all events
; ShowQueuedEvents = no              ; default: no
; ShowQueuedUnchanged = no           ; default: no
; ShowSelfSchedulingEvents = no      ; default: no
; ShowQueuedSelfSchedulingEvents = no ; default: no
;
; attributes
;
; ShowAttributes = no                ; default: no
; SelectedAttributes = year,alive    ; comma separated list of attribute names, if empty all attributes
; MinimumAttribute = 1               ; double value, default: -inf
; MaximumAttribute = 1               ; double value, default: +inf
;
; table increments
;
; ShowTableIncrements = no           ; default: no
; SelectedTables = t1,t2,t3          ; comma separated list of table names, if empty all tables
```

[back to topic contents]

Event trace options - format

Option	Type	Default	Description
ReportStyle	string	`modgen`	One of `readable`, `csv`, or `modgen` (see below in this section).
MaximumLines	integer	20000	Blocks messages if the line count exceeds this value. A final message is written to the trace.
NameColumnWidth	integer	40	The width of the `Name` column (column 7) in message output.

The `ReportStyle` option specifies the content and format of trace output. It can be `readable`, `csv`, or `modgen`. For compatibility for x-compatible models, the default value is `modgen`.

The examples earlier in this topic illustrate the `readable` style.

The `csv` style has almost the same content and layout as the `readable` style, except transformed into `csv` format for use by downstream

applications such as Excel. The `csv` style contains an additional leading column `Line` which contains the original line number. This can be useful as a secondary sort key to disambiguate the order of otherwise tied records. In the example below, lines 8 and 9 have identical `Time` but `Line` indicates the order in which the two attributes changed value. The exact order can be particularly important if events have tied times, or if an attribute changes value more than once at a given time.

In `csv` format some columns may have additional numeric precision compared to `readable` format.

The trace file name extension can be changed from the default `.txt` to `.csv` using the `OpenM.TraceFilePath` model run option, e.g. in a run `ini` file. See Trace file options.

Here is the output for the previous example Attributes with event context using `ReportStyle = csv`:

```
Line,Time,Entity,Age,Id,Trace,Value,Name,Remarks
1,0,"Person",0,222,"ENTER",,,
2,0,"Person",0,222,"attr",0,"in_union","initial"
3,0,"Person",0,222,"attr",0,"union_duration","initial"
4,18.19845239664,"Person",18.19845239664,222,"EVENT",,"Union1FormationEvent",
5,18.19845239664,"Person",18.19845239664,222,"attr",1,"in_union","was 0"
6,19.19845239664,"Person",19.19845239664,222,"attr",1,"union_duration","was 0"
7,19.94488519046,"Person",19.94488519046,222,"EVENT",,"Union1DissolutionEvent",
8,19.94488519046,"Person",19.94488519046,222,"attr",0,"in_union","was 1"
9,19.94488519046,"Person",19.94488519046,222,"attr",0,"union_duration","was 1"
10,20.01096416538,"Person",20.01096416538,222,"EVENT",,"Union2FormationEvent",
11,20.01096416538,"Person",20.01096416538,222,"attr",1,"in_union","was 0"
12,20.21281842274,"Person",20.21281842274,222,"EVENT",,"FirstPregEvent",
13,21.01096416538,"Person",21.01096416538,222,"attr",1,"union_duration","was 0"
14,21.19845239664,"Person",21.19845239664,222,"EVENT",,"UnionPeriod2Event",
15,23.01096416538,"Person",23.01096416538,222,"attr",2,"union_duration","was 1"
16,25.01096416538,"Person",25.01096416538,222,"attr",3,"union_duration","was 2"
17,29.01096416538,"Person",29.01096416538,222,"attr",4,"union_duration","was 3"
18,33.01096416538,"Person",33.01096416538,222,"attr",5,"union_duration","was 4"
19,100,"Person",100,222,"EVENT",,"DeathEvent",
20,100,"Person",100,222,"EXIT",,,
```

Here is the same output, displayed in a table as it might look in Excel:

Line	Time	Entity	Age	Id	Trace	Value	Name	Remarks
1	0	Person	0	222	ENTER			
2	0	Person	0	222	attr	0	in_union	initial
3	0	Person	0	222	attr	0	union_duration	initial
4	18.19845239664	Person	18.19845239664	222	EVENT		Union1FormationEvent	
5	18.19845239664	Person	18.19845239664	222	attr	1	in_union	was 0
6	19.19845239664	Person	19.19845239664	222	attr	1	union_duration	was 0
7	19.94488519046	Person	19.94488519046	222	EVENT		Union1DissolutionEvent	
8	19.94488519046	Person	19.94488519046	222	attr	0	in_union	was 1
9	19.94488519046	Person	19.94488519046	222	attr	0	union_duration	was 1
10	20.01096416538	Person	20.01096416538	222	EVENT		Union2FormationEvent	
11	20.01096416538	Person	20.01096416538	222	attr	1	in_union	was 0
12	20.21281842274	Person	20.21281842274	222	EVENT		FirstPregEvent	
13	21.01096416538	Person	21.01096416538	222	attr	1	union_duration	was 0
14	21.19845239664	Person	21.19845239664	222	EVENT		UnionPeriod2Event	
15	23.01096416538	Person	23.01096416538	222	attr	2	union_duration	was 1
16	25.01096416538	Person	25.01096416538	222	attr	3	union_duration	was 2
17	29.01096416538	Person	29.01096416538	222	attr	4	union_duration	was 3
18	33.01096416538	Person	33.01096416538	222	attr	5	union_duration	was 4

Line	Time	Entity	Age	Id	Trace	Value	Name	Remarks
19	100	Person	100	222	EVENT		DeathEvent	
20	100	Person	100	222	EXIT			

The `modgen` output style looks like this:

```
Person - actor_id=1 - case_seed=1 -  : event=timeDeathEvent - time=100.000000000000000
Person - actor_id=1 - case_seed=1 -  : event=timeFirstPregEvent - time=inf
Person - actor_id=1 - case_seed=1 -  : event=timeUnion1DissolutionEvent - time=inf
Person - actor_id=1 - case_seed=1 -  : event=timeUnion1FormationEvent - time=inf
Person - actor_id=1 - case_seed=1 -  : event=timeUnion2DissolutionEvent - time=inf
Person - actor_id=1 - case_seed=1 -  : event=timeUnion2FormationEvent - time=inf
Person - actor_id=1 - case_seed=1 -  : event=timeUnionPeriod2Event - time=inf
Person - actor_id=1 - case_seed=1 -  : event=scheduled - 0 - time=1.000000000000000
Person - actor_id=1 - case_seed=1 -  : event=timeDeathEvent - time=100.000000000000000
Person - actor_id=1 - case_seed=1 -  : event=scheduled - 0 - time=2.000000000000000
Person - actor_id=1 - case_seed=1 -  : event=timeDeathEvent - time=100.000000000000000
...
```

The `modgen` report style allows detailed comparison of modgen and ompp versions of a x-compatible model. The Test Models utility rearranges, reformats, and normalizes `modgen` style of event trace output of both Modgen and OpenM++ versions of a model to improve comparability and help understand differences.

[back to event trace options]
[back to topic contents]

Event trace options - filters

Filter options pass or block *any* kind of event trace message based on the characteristics of the entity when it produced the message.

Option	Type	Default	Description
SelectedEntityKinds	comma separated list	empty	Block any message if entity kind (e.g. `Person`) not in this list (unless list is empty).
SelectedEntities	comma separated list	empty	Block any message if entity not in this list (unless list is empty).
SelectLinkedEntities	yes/no	no	An active entity in `SelectedEntities` will add new linked entities to `SelectedEntities`.
SelectedCaseSeeds	comma separated list	empty	Block any message if case seed not in this list (unless list is empty).
MinimumTime	floating point number	-inf	Block any message if global time < this.
MaximumTime	floating point number	+inf	Block any message if global time > this.
MinimumAge	floating point number	-inf	Block any message if entity age < this.
MaximumAge	floating point number	+inf	Block any message if entity age > this.

For examples, see Worked example 1 Using filters

[back to event trace options]
[back to topic contents]

Event trace options - events

Event options pass or block specific kinds of message associated with entity lifecycle and events.

Option	Type	Default	Description

Option	Type	Default	Description
ShowEnterSimulation	yes/no	yes	Pass/block entity entrance message.
ShowExitSimulation	yes/no	yes	Pass/block entity exit message.
ShowEvents	yes/no	yes	Pass/block event occurrence message.
SelectedEvents	comma separated list	empty	Block event message if event not in this list (unless list is empty).
ShowQueuedEvents	yes/no	no	Pass/block event time message.
ShowQueuedUnchanged	yes/no	no	Pass/block event time message if future time did not change.
ShowSelfSchedulingEvents	yes/no	no	Pass/block self-scheduling event occurrence message.
ShowQueuedSelfSchedulingEvents	yes/no	no	Pass/block self-scheduling event time message.

An entity produces a message when it recalculates the future scheduled time of an event. However, the default setting `ShowQueuedUnchanged = no` blocks the message if the recalculated future scheduled time did not change.

For examples, see Worked example 1 Using filters

[back to event trace options]
[back to topic contents]

Event trace options - attributes

Attribute options pass or block messages for attributes, links, and multilinks. An entity produces a message for an attribute when it enters the simulation and whenever the attribute changes.

Option	Type	Default	Description
ShowAttributes	yes/no	no	Pass/block attribute message
SelectedAttributes	comma separated list	empty	Block attribute message if attribute not in this list (unless list is empty).
MinimumAttribute	floating point number	-inf	Block attribute message if attribute value < this.
MaximumAttribute	floating point number	+inf	Block attribute message if attribute value > this value.

If `SelectedAttributes` is not specified, *all* attribute messages are passed, including for attributes created by the OpenM++ compiler.

`MinimumAttribute` and `MaximumAttribute` apply only to normal numerical attributes, not links or multilinks.

The value of an attribute of type `bool` is 0 for `false` and 1 for `true`. The value of an attribute of type classification or partition is `{0,1,2,...}`.

For examples, see
Worked example 2 Tracing attributes
Worked example 3 Tracing links and multilinks

[back to event trace options]
[back to topic contents]

Event trace options - table increments

Table increment options pass or block messages related to pushing table increments to accumulators. An entity produces a table increment message whenever it pushes a completed increment to an accumulator.

Option	Type	Default	Description
ShowTableIncrements	yes/no	no	Pass/block push increment message
SelectedTables	comma separated list	empty	Block table increment message if table not in this list (unless list is empty).

If `SelectedTables` is not specified, *all* push increment messages are passed.

For an example, see Worked example 4.

[back to event trace options]
[back to topic contents]

Trace file options

A number of `OpenM` options control if, how, and where the trace file is produced. The following is extracted from `OM_ROOT/props/model/Model-example.ini`.

```
;# 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
```

[back to topic contents]

Trace file API

Trace output can be toggled off or on from model code provided the model was built with event trace capability. The API consists of the two functions

```
void StartEventTrace(void);
void StopEventTrace(void);
```

This can be helpful in situations where the filtering functionality described in this topic is insufficient. For example, a rare condition in model code can be used to toggle event trace on to identify entities which experienced the condition for further exploration. Event trace is on at the beginning of a run, so an initial call to `StopEventTrace` at the beginning of a run (or case) is required for a subsequent call to `StartEventTrace` to have an effect.

[back to topic contents]

External Names

This topic describes how to specify the names exposed to external software for parameter and table dimensions, table expressions, and enumerations of classifications. These names are used in `csv` files produced by `dbcopy` and in downloads from the OpenM++ UI.

Related topics

- Model Code

Topic contents

- Default name
- Explicit name
- Identifying missing explicit names
- Heuristic name
- Name restrictions
- All generated names

Default name

Each dimension of a parameter or table, each expression of a table, and each enumerator of a classification has an associated *external name*. The OpenM++ compiler provides a *default name* for each *external name*. The *default name* can be overridden by an *explicit name* or a *heuristic name*.

The default name for a parameter dimension has the form `ParameterName.DimN`, where `N` is `{0,1,...,rank-1}`, and `rank` is the number of parameter dimensions.

The default name for a table dimension has the form `TableName.DimN`, where `N` is `{0,1,...,rank-1}`, and `rank` is the number of classificatory dimensions in the table, i.e. the number of dimensions not counting the expression 'dimension'. Because `rank` excludes the expression dimension of a table, the expression dimension is skipped over in the numbering of a table dimension default name.
Note: The default name of a table dimension may differ from what's used to identify a dimension in `//LABEL` and `/*NOTE` documentation comments. For compatibility with Modgen models, documentation comments count the expression dimension of the table in the numbering scheme. For more on labels and notes, see symbol labels.

The default name for a table expression has the form `TableName.ExprN`, where N is `{0,1,...,expressions-1}`, and `expressions` is the number of expressions in the table.

The default name of an enumerator of a classification is the same as the enumerator name in model code.

Here's an example of the default names for a 1-dimensional table:

```
table Person T02_TotalPopulationByYear  //EN Life table
{
   //EN Curtate age
   integer_age *
   {
      unit,       //EN Population start of year
      duration()  //EN Average population in year
   }
};
```

This table has a single classificatory dimension with *default name* `Dim0` and two expressions with default names `Expr0` and `Expr1`. These names identify individual cells in the table. If a run with this table is exported in `.csv` format, an extract of the file `T02_totalPopulationByYear.csv` might look like this:

expr_name	Dim0	expr_value
Expr0	0	5000
Expr0	1	5000

expr_name	Dim0	expr_value
Expr0	2	5000
...		
Expr0	99	5000
Expr0	100	5000
Expr1	0	5000
Expr1	1	5000
Expr1	2	5000
...		

Where `Dim0` identifies the cell coordinates and `Expr0` and `Expr1` identify the expression. The generated default names `Dim0`, `Expr0`, and `Expr1` are positional, not descriptive. That can make downstream use of exported results difficult and error-prone.

Here's an example of the default names of the enumerators of a classification, taken from the `RiskPaths` model. The `UNION_ORDER` classification has the following declaration:

```
classification UNION_ORDER  //EN Union order
{
   UO_FIRST,    //EN First union
   UO_SECOND    //EN Second union
};
```

The default names of the two enumerators are the same as the codes in the declaration: `UO_FIRST` and `UO_SECOND`.

[back to topic contents]

Explicit name

An *explicit name* can be assigned to dimension and expressions in model source code using the naming operator `=>`, in which case it replaces the default name. The following example replaces the default names `Dim0`, `Expr1`, and `Expr2` with more descriptive names:

```
table Person T02_TotalPopulationByYear //EN Life table
{
   //EN Curtate age
   age => integer_age *
   {
      pop => unit,     //EN Population start of year
      py => duration() //EN Average population in year
   }
};
```

The table dimension is now named `age` and the measures are named `pop` and `py`. The `.csv` file would now look something like:

expr_name	age	expr_value
pop	0	5000
pop	1	5000
pop	2	5000
...		
pop	99	5000
pop	100	5000
py	0	5000
py	1	5000
py	2	5000

expr_name	age	expr_value
...		

An explicit name can also be specified with a comment-based syntax using the default name. The following lines have the same effect as the preceding example:

```
//NAME T02_TotalPopulationByYear.Dim0 age
//NAME T02_TotalPopulationByYear.Expr0 pop
//NAME T02_TotalPopulationByYear.Expr1 py
```

Modgen-specific: The naming operator `=>` is not recognized by Modgen and will produce a syntax error. For x-compatible model code, use the `//NAME` syntax.

Explicit names can be specified for dimensions of a parameter. For example, the parameter declaration

```
double UnionDurationBaseline[UNION_ORDER][UNION_DURATION];
```

can incorporate explicit names using the naming operator before a dimension:

```
double UnionDurationBaseline
   Order    => [UNION_ORDER]
   Duration => [UNION_DURATION];
```

or by using the comment-based syntax:

```
//NAME UnionDurationBaseline.Dim0 Order
//NAME UnionDurationBaseline.Dim1 Duration
```

Explicit names can be specified for an enumerator of a classification. For the classification declaration

```
classification UNION_ORDER  //EN Union order
{
   UO_FIRST,    //EN First union
   UO_SECOND    //EN Second union
};
```

an explicit name can be specified using the naming operator before the enumerator:

```
classification UNION_ORDER  //EN Union order
{
   First  => UO_FIRST,       //EN First union
   Second => UO_SECOND       //EN Second union
};
```

or by using the comment-based syntax

```
//NAME UO_FIRST First
//NAME UO_SECOND Second
```

[back to topic contents]

Identifying missing explicit names

If a default name is being used, a downloaded parameter or table has column names like `Dim0` or `Dim2`, and table expressions like `Expr2` or `Expr5`, which are less than helpful for model users. An issue for model developers is to identify missing explicit names like these, and, once identified, to insert the missing explicit name in the model code.

The OpenM++ compiler supports a family of options to aid that process. Each member of the family targets a specific kind of missing explicit name. When an option is set to `on`, the compiler will generate a warning for each missing explicit name of that kind. The warning includes the model code file and line where the symbol was declared. In an IDE like Visual Studio, double-clicking on the warning in the log window navigates immediately to that model source code location in the IDE editor.

By default these options are `off`. Multiple options can be turned `on` at the same time.

The following example identifies all dimensions and expressions of published tables in `RiskPaths` which lack an explicit name. Inserting the following line in `ompp_framework.ompp`

```
options missing_name_warning_published_table = on;
```

causes the compiler to emit warnings like:

```
1>../code/Tables.mpp(40): warning : missing explicit name for dimension 0 of published table 'T02_TotalPopulationByYear'
1>../code/Tables.mpp(42): warning : missing explicit name for expression 0 of published table 'T02_TotalPopulationByYear'
1>../code/Tables.mpp(43): warning : missing explicit name for expression 1 of published table 'T02_TotalPopulationByYear'
```

Double-clicking one of these warnings navigates directly to the model code line of the dimension or expression.

The following table lists the available options to emit warnings for missing explicit names, grouped by category. The Scope column shows what produces a warning for the given option.

Option	Scope
All	
missing_name_warning_classification	classification level (enumerator)
missing_name_warning_parameter	dimension
missing_name_warning_table	dimension, expression
Published only	
missing_name_warning_published_classification	as above, but only for published symbols
missing_name_warning_published_parameter	as above, but only for published symbols
missing_name_warning_published_table	as above, but only for published symbols

[back to topic contents]

Heuristic name

A *heuristic name* is a name which replaces a default name with a name generated by the OpenM++ compiler. A heuristic name is generated based on contextual information about the dimension or expression, if no explicit name was provided in model code. Explicit names are generally preferable to heuristic names. Heuristic names can provide an immediate improvement in the usability of downloaded parameters and tables, replacing default names like `Dim2` or `Expr5` with something better.

Heuristic names are not generated by default. To generate heuristic names, include the following statement in model source code:

```
options use_heuristic_short_names = on;
```

The table in the previous example, with no explicit names, would produce the following exported `csv`:

expr_name	Curtate_age	expr_value
Population_start_of_year	0	5000
Population_start_of_year	1	5000
Population_start_of_year	2	5000
...		
Population_start_of_year	99	5000
Population_start_of_year	100	5000
Average_population_in_year	0	5000
Average_population_in_year	1	5000

expr_name	Curtate_age	expr_value
Average_population_in_year	2	5000
...		

In this example, the OpenM++ compiler generated heuristic names using `//EN` labels found in the model source code (`EN` is the default language of this model). However, the OpenM++ compiler may, particularly if a label exceeds the name length limit, create a heuristic name based on other information, such as the name of the classification underlying the dimension of a table. To respect name length limits, a heuristic name may be based on a label with an interior portion snipped out and replaced by `_X_`, or prefixed by `X_` so that the name starts with an alphabetic character.

If a heuristic name clashes with the name of a previous dimension or measure, a disambiguating suffix will be appended to the heuristic name. For example the parameter `k_year` declared as

```
parameters {
   YEAR k_year[REGION][REGION];
};
```

has a repeated dimension `REGION`. The heuristic name for the second repeated dimension of `k_year` will be disambiguated by appending `Dim1`:

```
// Parameter k_year: k_year
//NAME k_year.Dim0 Region
//NAME k_year.Dim1 RegionDim1
```

The maximum length of heuristic names can be controlled by the following option:

```
options short_name_max_length = 32;
```

Heuristic name generation for enumerators of classifications can be disabled by the following option:

```
options enable_heuristic_names_for_enumerators = off;
```

By default, this option is `on`. If this option is `off`, the name of a classification enumerator will always be the same as the enumerator model code name. This option has effect only if the option `use_heuristic_short_names` is `on`.

[back to topic contents]

Name restrictions

Dimension and measure names in exported files facilitate direct use in downstream analysis. For example, a `.csv` could be opened in Excel and used as a pivot table, or imported into R or SAS, with meaningful column names. A wide variety of applications can be used to do downstream analysis, each with its own name restrictions. OpenM++ imposes the following restrictions to explicit names to reduce potential problems in downstream analysis:

A name

- has a maximum length in characters given by the option `short_name_max_length` (default 32)
- has characters in uppercase A-Z, lowercase a-z, digits 0-9, and the _ character
- is unique within the dimensions or expressions of the parameter or table

If a name does not meet these restrictions, the OpenM++ compiler will emit a warning and 'mangle' the name to meet the restrictions, e.g. by replacing forbidden characters by _, by truncating the name, or by appending a trailing numeric suffix to disambiguate identical names.

If `Default` values for a parameter are provided using a `.csv` file, any name used in the file must correspond to the corresponding *external name* in the model. The same applies to uploads of parameter data in `.csv` files, or to parameters supplied programmatically using an external script.

[back to topic contents]

All generated names

Any name generated or modified by the OpenM++ compiler is written to a file named `GeneratedNames.ompp` in the compiler output directory, which

in Windows is `MODEL/ompp/src/GeneratedNames.ompp`. `GeneratedNames.ompp` does not contain explicit names given in model source code using `=>` or `//NAME`, unless for some reason the OpenM++ compiler needed to modify them.

The content of `GeneratedNames.ompp` uses `//NAME` statements to make it suitable as a starting point to specify explicit names in model source code, for example in a separate source code module named `code/ExplicitNames.ompp`, or perhaps immediately following the declaration of a classification, parameter or table.

Here is an extract of `src/GeneratedNames.ompp` from the `RiskPaths` model:

```
// Parameter AgeBaselineForm1: Age baseline for first union formation
//NAME AgeBaselineForm1.Dim0 X_2_5_year_age_intervals

// Parameter AgeBaselinePreg1: Age baseline for first pregnancy
//NAME AgeBaselinePreg1.Dim0 X_2_5_year_age_intervals

// Parameter ProbMort: Death probabilities
//NAME ProbMort.Dim0 Simulated_age_range

// Table T01_LifeExpectancy: Life Expectancy
//NAME T01_LifeExpectancy.Expr0 Total_simulated_cases
//NAME T01_LifeExpectancy.Expr1 Total_duration
//NAME T01_LifeExpectancy.Expr2 Life_expectancy

// Table T02_TotalPopulationByYear: Life table
//NAME T02_TotalPopulationByYear.Dim0 Curtate_age
//NAME T02_TotalPopulationByYear.Expr0 Population_start_of_year
//NAME T02_TotalPopulationByYear.Expr1 Average_population_in_year

// Table T04_FertilityRatesByAgeGroup: Fertility rates by age group
//NAME T04_FertilityRatesByAgeGroup.Dim0 Age_interval
//NAME T04_FertilityRatesByAgeGroup.Dim1 Union_Status
//NAME T04_FertilityRatesByAgeGroup.Expr0 Fertility
```

Generated Model Documentation

This topic describes one of the two components of model documentation: the generated Symbol Reference.

This topic is under construction and/or revision.

Related topics

- Model Documentation: Home topic on model documentation
- Authored Model Documentation: The autonomous authored component of model documentation
- Model Symbols: Symbols in model code and in the user interface
- Symbol Labels and Notes: Human-language labels and notes for model symbols

Topic contents

- Introduction and outline
- Symbol Reference - User Edition
- Customizing the Symbol Reference
- Symbol Reference - Developer Edition
- Navigation Aid Topics

Introduction and outline

Model Documentation has two components: An autonomous authored component and a generated Symbol Reference component. This topic describes the latter.

Human-language labels and notes for model symbols are combined with symbol metadata and cross-reference information to produce a Symbol Reference for a model. There are two versions of the Symbol Reference, one targeted to model users and one to model developers. Each version can be customized using `options` statements in model code.

The User Edition of the Symbol Reference is targeted to model users. It documents all symbols exposed to users and deliberately excludes aspects of a model not exposed to users. Here are some examples of aspects excluded in the User Edition:

- Parameters deliberately suppressed from the UI using `parameters_retain`.
- Derived parameters
- Enumerations used only in model code
- Entity Table filters
- Attributes not exposed to users
- Modules
- cross-reference information on symbol use in model code

The Developer Edition of the Symbol Reference is targeted to model developers. It includes all content in the User Edition, and adds symbols, topics and topic sections for aspects of a model not exposed to users. For example, the Developer Edition includes a topic which lists all model code modules, as well as a topic for each module.

Independent of the Symbol Reference, labels and notes for all symbols exposed in the model UI are published to the model database, for all human languages declared in the model.

The Symbol Reference component of Model Documentation can be disabled by an option in model code:

```
options generated_documentation = off;
```

[back to topic contents]

Symbol Reference - User Edition

To explore the User Edition of the Symbol Reference for `RiskPaths`, turn off Developer Edition by commenting the following line in `RiskPaths/code/ompp_options.ompp` so that it looks like this:

```
//options symref_developer_edition = on;
```

Next, rebuild the model, run it, click the 'book' icon, and from the `RiskPaths Home` topic click `Symbol Reference`.

The browser should look like this:

By default, a model's Symbol Reference documents only symbols which are exposed to users. Labels and notes for model symbols which are not available to users when the model is run are not published, and are not present in the Symbol Reference. This includes parameters and tables suppressed from the model at build time and associated classifications, ranges, and partitions not used by other published symbols. Information on attributes is published only if the model was built with microdata output enabled using the statement:

```
options microdata_output = on;
```

Symbol Reference - Developer Edition

To explore the Developer Edition of the Symbol Reference for `RiskPaths`, turn on Developer Edition by uncommenting the following line in `RiskPaths/code/ompp_options.ompp` so that it looks like this:

```
options symref_developer_edition = on;
```

Next, rebuild the model, run it, click the 'book' icon, and from the `RiskPaths Home` topic click `Symbol Reference`.

The browser should look like this:

The Developer Edition of the Symbol Reference augments the User Edition as follows:

- Topics for unpublished parameters, tables, attributes, and enumerations.
- Cross-reference tables in each symbol topic.
- Module topics

[back to topic contents]

Customizing the Symbol Reference

Options which suppress the main topic:

Option	Default	Effect if changed from default
symref_main_topic	on	Removes the main topic including the navigation aid table.
symref_model_symbol	on	Removes the topic for the unique model symbol, if declared in the model.

Options which suppress navigation aids:

Option	Default	Effect if changed from default
symref_parameters_alphabetic	on	Removes the parameters alphabetic list topic.
symref_parameter_major_groups	on	Removes the parameter major groups topic.
symref_parameter_hierarchy	on	Removes the parameter hierarchy topic and parameter hierarchy breadcrumbs.
symref_tables_alphabetic	on	Removes the tables alphabetic list topic.
symref_table_major_groups	on	Removes the table major groups topic.
symref_table_hierarchy	on	Removes the table hierarchy topic and table hierarchy breadcrumbs.
symref_attributes_alphabetic	on	Removes the attributes alphabetic list topic.
symref_enumerations_alphabetic	on	Removes the enumerations alphabetic list topic.

Options which suppress individual topics:

Option	Default	Effect if changed from default
symref_topic_parameters	on	Removes parameter topics and parameter cross-reference sections in other topics.
symref_topic_tables	on	Removes table topics and table cross-reference sections in other topics.
symref_topic_attributes	on	Removes attribute topics.
symref_topic_enumerations	on	Removes enumeration topics and enumeration cross-reference sections in other topics.

Options controlling symbol notes:

Option	Default	Effect if changed from default
symref_topic_notes	on	Removes the note sections of *all* symbol topics.
symref_topic_notes_early	on	Places notes after other sections in symbol topics.
symref_topic_note_heading	on	Suppresses the Note: heading in symbol topics.

Options affecting only the Developer Edition:

Option	Default	Effect if changed from default
symref_developer_edition	off	Turns on symbols not exposed to users and appends "(Developer Edition)" to title in main topic.

Option	Default	Effect if changed from default
symref_unpublished_symbols	on	Removes unpublished symbols, e.g. those absent due to parameters_retain or tables_retain statements in model code.
symref_global_functions	on	Removes global function cross-reference sections in symbol topics.
symref_entity_functions	on	Removes entity function cross-reference sections in symbol topics.
symref_identity_attributes	on	Removes identity attribute cross-reference sections in symbol topics.
symref_topic_entity_sets	on	Removes entity set alphabetic list, topics, and cross-reference sections in symbol topics.
symref_topic_modules	on	Removes module alphabetic list, topics, and cross-reference sections in symbol topics.
symref_topic_modules_use	off	Includes use module topics.
symref_topic_modules_symbols_declared	on	Turns off the alphabetic "symbols declared" section in individual module topics.
symref_topic_attributes_internal	on	Turns off attribute topics for internally-generated attributes, including derived attributes.

[back to topic contents]

Navigation Aid Topics

The following table lists navigation aid topics in the Symbol Reference.

Navigation aid	Topic name	Description and notes
Symbol Reference	symbol-reference	The Symbol Reference main topic.
Parameters in alphabetic order	parameters-alphabetic	
Parameter hierarchy	parameter-hierarchy	Present if the model has at least one published parameter group. May contain a trailing 'Orphan parameters' group.
Parameter major groups	parameter-major-groups	Present if parameter hierarchy present.
Tables in alphabetic order	tables-alphabetic	
Table hierarchy	table-hierarchy	Present if the model has at least one published table group. May contain a trailing 'Orphan tables' group.
Table major groups	table-major-groups	Present if table hierarchy present.
Enumerations in alphabetic order	enumerations-alphabetic	
Modules in alphabetic order	modules-alphabetic	
Attributes in alphabetic order	attributes-alphabetic	
Entity sets in alphabetic order	entity-sets-alphabetic	

Note that derived parameters published as tables group with tables in the navigation aid topics, not with parameters.

[back to topic contents]

Groups

Groups are hierarchical collections of parameters, tables and attributes declared in model code.

Related topics

- OpenM++ User Interface
- Model Code

Topic contents

- Parameter groups Organizing parameters
- Table groups Organizing tables
- Attribute groups Organizing entity attributes

Parameter groups

A parameter group is a named, ordered list of parameters and other parameter groups, e.g.

```
parameter_group PG12_SchoolOneFate  //EN Primary School
{
   Educ1Model,
   PG10_SchoolOneFateBase,
   PG11_SchoolOneFateRefined,
   PG10_ShoolOneTracking
};
```

Parameter groups are described in a wiki subtopic here .

[back to topic contents]

Table groups

A table group is a named, ordered list of tables and other table groups, e.g.

```
table_group TG04_Education //EN Education
{
   TG04_Preschool,
   TG04_Primary,
   TG04_Secondary
};
```

Table groups are described in a wiki subtopic here .

[back to topic contents]

Attribute groups

An attribute group is a named, ordered list of attributes of a specified kind of entity, and other attribute groups.

The following is an example of an attribute group declaration for the `RiskPaths` model:

```
attribute_group Person AG02_Unions  //EN Union-related attributes
{
   dissolution_duration,
   in_union,
   union_duration,
   union_period2_change,
   union_status,
   unions
};
```

The name of a model code module, quoted, can also be an element of an attribute group. It will be expanded into a list of all the attributes of the specified kind declared in the module, in lexicographical order. For example, the following attribute group

```
attribute_group Person AG03_Unions  //EN Union-related attributes
{
   "Unions.mpp"
};
```

has exactly the same attributes, and in the same order, as the attribute group `AG02_UnionAttributes` shown above.

Attribute groups can be used to specify attributes in an `ini` file or a command-line option, e.g.

```
[Microdata]
ToCsv = yes
ToDB = yes
Person = CommonAttributes, union_status
```

where the attribute group `CommonAttributes` is declared in model code as

```
attribute_group Person CommonAttributes  //EN Commonly-used attributes
{
   case_id,
   entity_id,
   age
};
```

[back to topic contents]

Illustrative Model `Align1`

`Align1` is an experimental model which manipulates the event queue to align with external counts. This topic describes the approach and implementation, and includes some experiments and notes.

Related topics

- Model Code

Topic contents

- Introduction and description
- Experiment #1 Illustrative run
- Experiment #2 Computational cost
- Remarks Some notes on the approach and possible future steps
- `Align1` code Model code module `Alignment.mpp`
- `Align1` input Default parameter values `Alignment.dat`

Introduction and description

The `Align1` model is a proof of concept and testbed for dynamic alignment of time-based models using event queue look-ahead. `Align1` steers itself to aggregate annual targets by reading and modifying the event queue dynamically at the beginning of each year.

`Align1` is based on the `NewTimeBased` model which is part of the OpenM++ distribution. It adds alignment apparatus in a new module `Alignment.mpp` and associated parameters in `Alignment.dat` but is otherwise unmodified. `NewTimeBased` has a `Ticker` entity with a timekeeping `TickEvent` and `Person` entities with `Mortality` events. `Align1` adds a new event `AlignmentEvent` to `Ticker`, which occurs at the same time as `TickEvent` but at lower priority. The input parameter `MortalityAlignmentTarget` contains target mortality counts by year. At the beginning of each year `AlignmentEvent` reads the event queue and counts the number of deaths scheduled to occur during that year. If the count is higher than the target for the year, enough scheduled events are deferred to the subsequent year to hit the target. If the count is lower than the target, enough scheduled events are advanced from subsequent years to the current year to hit the target. The exact rules are mechanical and described in model code comments. This process is repeated at the beginning of each year.

The events which are deferred or advanced to hit alignment targets are those which are closest to the upper boundary of the current alignment year.

The Default run has 10,000 Persons and sets `MortalityAlignmentTarget` to the mortality counts which would occur in the absence of alignment (using table MortalityCounts from a previous run with alignment off).

[back to topic contents]

Experiment #1

This experiment sets target mortality to 122 in each of the 20 years. 122 is the average annual mortality in the 20 years in the Default run with no alignment. The first column shows mortality counts in the Default run with no alignment. One observes a secular decrease (with some noise) because the population is progressively smaller so fewer die each year due to the lower base population (the morality hazard is constant). The target column, in contrast, sets the number of deaths to a fixed value in each year. Compared to the Default run, deaths need to be decreased in early years and increased in later years to hit the target.

Year	Mortality (Default)	Target	Mortality (Aligned)	Events Deferred	Events Advanced
0	141	122	122	19	0
1	144	122	122	41	0
2	114	122	122	33	0
3	146	122	122	57	0
4	142	122	122	77	0

Year	Mortality (Default)	Target	Mortality (Aligned)	Events Deferred	Events Advanced
5	126	122	122	81	0
6	129	122	122	88	0
7	139	122	122	105	0
8	111	122	122	94	0
9	118	122	122	90	0
10	112	122	122	80	0
11	118	122	122	76	0
12	102	122	122	56	0
13	110	122	122	44	0
14	104	122	122	26	0
15	116	122	122	20	0
16	114	122	122	12	0
17	124	122	122	14	0
18	112	122	122	4	0
19	112	122	122	0	6

The algorithm attained the annual targets by deferring mortality events in each of the first 19 years and advancing 6 mortality events for the final year. Mortality events were advanced only in the final year of the run because for all other years sufficient deaths had been deferred in previous years. The annual targets are hit exactly because event times are not recalculated during the year-in-progress in this model.

[back to topic contents]

Experiment #2

This second experiment explored computational cost and scaling behaviour, using a run with 10 million `Person` entities. With this population size, there were over 100,000 mortality events per year. Three runs were done.

Run	Description	Time
1	No alignment	1m25s
2	Alignment with targets=actual	1m27s
3	Alignment with targets=+/- 5% actual	1m26s

Run 1 had alignment disabled. Run 2 had alignment enabled, but the targets were the same as the results without alignment. So, no adjustment of the event queue was done, but the event queue was probed each year. Run 3 had random targets within +/- 5% of the original mortality results. So alignment was doing some work with the event queue to hit the targets in run 3.

The table shows that the run times were indistinguishable. For this model, anyway, the incremental cost of alignment was barely detectable.

[back to topic contents]

Remarks

1. A natural way for continuous time models to align is to tinker with the timing of events which “would have occurred anyway”. This helps preserve aspects of model logic, since prohibited events remain prohibited under alignment.

2. An event which was deferred or advanced by alignment will still have its event time recomputed if entity attributes change (perhaps to +inf if the entity is no longer eligible for the event). This maintains aspects of the internal causative logic of the model, even under alignment.

3. The ‘event censoring’ optimization should perhaps not be used in models using this alignment technique, since that might deplete the pool of future events which can be advanced into an alignment interval. For example, in experiment 1 above, the 6 events which were advanced to

year 19 to hit the target had original times beyond the end of the run (which ended at time=20). They would have been right censored hence never placed in the queue.

4. In the current version of `Align1`, event times do not change during the simulation within an alignment window (year). It might be interesting to add a birthday event to the model and have a mortality schedule which varies by single year of age, to make tests more realistic.

5. The algorithm could be adapted to split an alignment interval (year) into sub-intervals (e.g. 10 equal intervals in each year), with recalculation of progress to the target for the current year. That would allow the algorithm to adjust for interacting events during the simulation of the current year. That would require counting events as they occur during the alignment interval, which was not needed in this version of `Align1`.

6. As the number of alignment targets increases, and if targets are classified by entity characteristics (e.g. age group), the code volume for alignment could become massive and error-prone (even though mechanical). That makes it a tempting candidate for new supporting functionality to automate some aspects.

[back to topic contents]

Model code

Below is the model source code for the module `Alignment.mpp`:

```
//LABEL (Alignment, EN) Alignment using event queue

#include "omc/optional_IDE_helper.h" // help an IDE editor recognize model symbols

#if 0 // Hide non-C++ syntactic island from IDE

parameters {
   bool EnableAlignment;
   int MortalityAlignmentTarget[REPORT_TIME];
};


actor Ticker
{
   //EN Time of next Alignment event
   TIME next_alignment;

   //EN Mortality events deferred by alignment (cumulative)
   int mortality_deferred;

   //EN Mortality events advanced by alignment (cumulative)
   int mortality_advanced;

   //EN Deficit in advanced mortality events because queue was exhausted (cumulative)
   int mortality_deficit;

   // AlignmentEvent should be lower priority than any other event
   // so that it executes after other tied events which might influence
   // events in the current alignment interval.
   event timeAlignmentEvent, AlignmentEvent, 1; //EN Alignment event
};

table Person MortalityCounts
{
   report_time
   * {
      entrances(alive, false) //EN Mortality events
   }
};

table Ticker AlignmentReport
{
   report_time
   * {
      mortality_deferred,     //EN Mortality events deferred
      mortality_advanced,     //EN Mortality events advanced
      mortality_deficit       //EN Mortality target deficit
   }
};

#endif // Hide non-C++ syntactic island from IDE

TIME Ticker::timeAlignmentEvent()
{
   // is synchronous with TickEvent, but lower priority
   return EnableAlignment ? next_alignment : time_infinite;
}

void Ticker::AlignmentEvent(void)
{
   // get event_id of MortalityEvent
```

```
int event_id_mortality = omr::event_name_to_id("MortalityEvent");
assert(event_id_mortality != -1); // MortalityEvent not found

// width of the target window
TIME alignment_window_width = 1.0;

// The upper bound of the target window, NB is just beyond the current alignment window
TIME alignment_window_upper_bound = time + alignment_window_width;

int alignment_window_target_count = MortalityAlignmentTarget[report_time];

//
// walk the event queue, from present to future
//
// add events to defer_list or advance_list
// as needed to hit the target count in the alignment time window
//
auto& event_queue = *BaseEvent::event_queue; // alias for the model event queue
std::forward_list<BaseEvent*> defer_list;    // list of events to defer
std::forward_list<BaseEvent*> advance_list;  // list of events to advance
int unadjusted_count = 0; // count of scheduled events in the alignment time window before alignment
int deferred_events = 0;  // count of scheduled events deferred to the future beyond the alignment time window
int advanced_events = 0;  // count of scheduled events advanced from the future to within the alignment time window
for (auto evt : event_queue) {
    int id = evt->get_event_id();
    if (id != event_id_mortality) {
        // not a mortality event, skip
        continue;
    }
    double evt_time = evt->event_time;
    if (evt_time < alignment_window_upper_bound) {
        // we are inside the target alignment time window
        // update the count of currently scheduled events within the window
        ++unadjusted_count;
        if (unadjusted_count > alignment_window_target_count) {
            // there is an excess of scheduled events within the alignment time window
            // so add this event to the defer list
            defer_list.push_front(evt);
            ++deferred_events;
        }
    }
    else {
        // we are beyond the alignment time window
        if (advanced_events + unadjusted_count >= alignment_window_target_count) {
            // no need to find more events to advance, have found what's needed
            // so stop queue walk
            break;
        }
        // there is a deficit of events within the alignment time window
        // so add this event to the advance list
        advance_list.push_front(evt);
        ++advanced_events;
    }
}

if (alignment_window_target_count > unadjusted_count && unadjusted_count - alignment_window_target_count != advanced_events) {
    // there were insufficient events in the queue beyond the alignment window to meet the target
    mortality_deficit += alignment_window_target_count - unadjusted_count - advanced_events; // for AlignmentReport
}

if (deferred_events > 0) {
    mortality_deferred += deferred_events; // for AlignmentReport
    assert(advanced_events == 0);
    for (auto evt : defer_list) {
        // defer this event
        // remove it from the event queue
        event_queue.erase(evt);
        // postpone the event time by one alignment interval
        evt->event_time += alignment_window_width;
        // re-insert it to the event queue
        event_queue.insert(evt);
    }
}
else if (advanced_events > 0) {
    mortality_advanced += advanced_events; // for AlignmentReport
    // advance this event
    for (auto evt : advance_list) {
        // advance this event
        // remove it from the event queue
        event_queue.erase(evt);
        // advance the event time by an integral number of alignment intervals
        // until it falls within the alignment window
        TIME new_time = evt->event_time;
        while (new_time >= alignment_window_upper_bound) {
            new_time -= alignment_window_width;
        }
        evt->event_time = new_time;
        // re-insert it to the event queue
```

```
            // re-insert it to the event queue
            event_queue.insert(evt);
        }
    }
    else {
        // nothing to do
    }

    {
        // schedule next alignment
        Time t = next_alignment + alignment_window_width;
        if (t >= SimulationEnd) {
            next_alignment = time_infinite;
        }
        else {
            next_alignment = t;
        }
    }
}
```

[back to topic contents]

Model input

Below is the Default model input parmeters in `Alignment.dat`:

```
parameters {
    bool EnableAlignment = true;
    int MortalityAlignmentTarget[REPORT_TIME] = {
        141, //141,
        144, //144,
        114, //114,
        146, //146.
        142, //142,
        126, //126,
        129,
        139,
        111,
        118,
        112,
        118,
        102,
        110,
        104,
        116,
        114,
        124,
        112,
        112,
    };
};
```

[back to topic contents]

Lifecycle Attributes

An OpenM++ option can create a pair of built-in attributes, `lifecycle_event` and `lifecycle_counter`, which can be used to probe events in the entity.

Related topics

- Model Code

Topic contents

- Activating lifecycle attributes
- Worked example 1 Activating lifecycle attributes in `RiskPaths`
- Lifecycle attribute maintenance
- Worked example 2 Event counts
- Worked example 3 Cross-tabulated event counts
- Worked example 4 Statistics at an event
- Worked example 5 Microdata output at an event
- Worked example 6 Event context for microdata output
- Worked example 7 Filtering microdata output by event

Activating lifecycle attributes

The lifecycle attributes are not present by default. They can be created for a specified kind of entity using a statement like

```
options lifecycle_attributes = Person;
```

This example assumes that the model has a `Person` entity. This statement causes the OpenM++ compiler to create, for the `Person` entity, the built-in attributes `lifecycle_event` and `lifecycle_counter`, and the supporting built-in classification `LIFECYCLE_Person`.

Lifecycle attributes are activated on an entity-by-entity basis. For example, if the model also had a `Ticker` entity, lifecycle attributes could be enabled for `Ticker` entities as well as `Person` entities by including a second `options` statement

```
options lifecycle_attributes = Ticker;
```

This statement would cause OpenM++ compiler to create, for the `Ticker` entity, the built-in attributes `lifecycle_event` and `lifecycle_counter`, and the supporting built-in classification `LIFECYCLE_Ticker`.

Model code should not contain clashing explicit declarations of lifecycle attributes or the supporting classification or enumerators. If it does, a build error will occur.

[back to topic contents]

Worked example 1

This section shows how to create and examine lifecycle attributes for the `RiskPaths` model. `RiskPaths` contains a single kind of entity, `Person`. Request the creation of lifecycle attributes for `Person` by adding the following statement to `ompp_options.ompp`, if not already present:

```
options lifecycle_attributes = Person;
```

Build the model, open the UI, click the book icon, then click Symbol Reference.
The browser should display:

Next, click the Attributes link under Lists.

The browser should display:

Scroll down if necessary or click the letter 'l' (lower case 'L') at the top of the Attributes list to see the two attributes `lifecycle_counter` and `lifecycle_events`.

Click the link for `lifecycle_counter`.

The browser should display:

The built-in attribute `lifecycle_counter` created by the OpenM++ compiler has type `short`, which is a C++ integer type. The Symbol Reference shows that `lifecycle_counter` is currently unused in `RiskPaths` model code.

Return to the Attributes list by clicking the browser back button, or by clicking the Symbol Reference link, then the link to the Attributes list as before.

Next, click the link for `lifecycle_event`.
The browser should display:

The built-in attribute `lifecycle_event` created by the OpenM++ compiler has type `LIFECYCLE_Person`, which is a classification created by the OpenM++ compiler to support `lifecycle_event` and its use in model code. Click the link `LIFECYCLE_Person` after **Type:**.
The browser should display:

`LIFECYCLE_Person` Lifecycle

Kind: classification **Size:** 9 **Values:** {0,...,8}

Enumerators of `LIFECYCLE_Person`:

External Name	Enumerator	Value	Label
`LC_Person_enter_simulation`	`LC_Person_enter_simulation`	0	enter simulation
`LC_Person_exit_simulation_external`	`LC_Person_exit_simulation_external`	1	exit simulation (external)
`LC_Person_DeathEvent`	`LC_Person_DeathEvent`	2	DeathEvent
`LC_Person_FirstPregEvent`	`LC_Person_FirstPregEvent`	3	FirstPregEvent
`LC_Person_Union1DissolutionEvent`	`LC_Person_Union1DissolutionEvent`	4	Union1DissolutionEvent
`LC_Person_Union1FormationEvent`	`LC_Person_Union1FormationEvent`	5	Union1FormationEvent
`LC_Person_Union2DissolutionEvent`	`LC_Person_Union2DissolutionEvent`	6	Union2DissolutionEvent
`LC_Person_Union2FormationEvent`	`LC_Person_Union2FormationEvent`	7	Union2FormationEvent
`LC_Person_UnionPeriod2Event`	`LC_Person_UnionPeriod2Event`	8	UnionPeriod2Event

Attributes of type `LIFECYCLE_Person`:

Entity	Name	Label
`Person`	`lifecycle_event`	Lifecycle event

[Symbol Reference]
[Home]

The classification `LIFECYCLE_Person` created by the OpenM++ compiler has 9 enumerators, all with the prefix `LC_Person_`.
This prefix is intended to minimize the chance of a clash with other symbols in model code. The `LC_` portion of the prefix is a mnemonic for 'lifecycle'.

The first enumerator `LC_Person_enter_simulation` notes when a `Person` entity enters the simulation.
The second enumerator `LC_Person_exit_simulation_external` notes when a `Person` entity exits the simulation not from an event in that `Person` entity, for example from an event in some other entity.
The remaining `7` enumerators note the occurrence of the `7` `Person` events in `RiskPaths`, in alphabetic order by event name.

This concludes worked example 1. Worked example 2 involves additional modifications to model code. To prepare for it, close the browser window for `RiskPaths` model documentation, the browser window for the `RiskPaths` UI, and the `oms` instance which supported the `RiskPaths` UI, if necessary.

[back to topic contents]

Lifecycle attribute maintenance

OpenM++ run-time code generated by the OpenM++ compiler maintains lifecycle attributes as follows:

1. When an entity is created via `new`, `lifecycle_counter` is `0` and `lifecycle_event` is `LC_Person_enter_simulation`.

2. When the entity enters the simulation by a call to `enter_simulation()`, `life_cycle_event` is assigned `LC_Person_enter_simulation` (no change), and `lifecycle_counter` is incremented (so it changes and has the value `1`).

3. When an event in the entity occurs, `life_cycle_event` is assigned the corresponding enumerator for the event, and `lifecycle_counter` is incremented.

4. Self-scheduling events, e.g. `self_scheduling_int(age)` have no effect on `life_cycle_event` or `life_cycle_counter`.

5. When an entity exits the simulation by a call to `exit_simulation()` in an event in that entity, the value of `life_cycle_event` and `life_cycle_counter` retain the values for that event, as in 3 above. If the call to `exit_simulation()` comes from elsewhere, for example from an event in some other entity, `life_cycle_event` is assigned `LC_Person_exit_simulation_external` and `lifecycle_counter` is incremented.

This means that

1. `lifecycle_counter` always changes whenever an event occurs in the entity.
2. `lifecycle_event` contains the enumerator corresponding to that event.
3. When an entity exits the simulation, `lifecycle_event` contains either the event responsible for the exit, or `LC_Person_exit_simulation_external` if the exit was not caused by an event in the entity.

Note that an event can occur with no change to `lifecycle_event` if the same event occurs in succession. But even if the same event occurs in succession, `lifecycle_counter` will change. So, for example, the self-scheduling derived attribute `trigger_changes(lifecycle_counter)` is guaranteed to note all events in the entity, whereas `trigger_changes(lifecycle_event`) may not.

Entity tables process increments only after all attribute changes in an event have completed. So, attribute values tabulated using `lifecycle_counter` or `lifecycle_event` reflect the values of attributes *after* all event-related model code has executed.

In models with entity links, an event in entity A can change the attributes of a different entity B. However, the value of `lifecycle_counter` and `lifecycle_event` in entity B do not change due to the event in entity A. They retain the values they had at the most recent event which occurred in entity B.

[back to topic contents]

Worked example 2

This example shows how to tabulate event counts by event.

Ensure that lifecycle attributes are enabled by the following statement in model code:

```
options lifecycle_attributes = Person;
```

In `RiskPaths`, copy/paste the following code to `code/Tables.mpp`:

```
table snapshot untransformed Person X01_Events //EN All Events
[ trigger_changes(lifecycle_counter) ]
{
   lifecycle_event //EN Event
 * {
       unit //EN N
   } //EN Statistic
};
```

Build and run the model, selecting the output table `X01_Events`.

The browser should display:

The filter of this snapshot table `trigger_changes(lifecycle_counter)` is instantaneously `true` when any event occurs, and is `false` otherwise. The analysis dimension consists of the single expression `unit`, which sums increments whose values are always `1`. The table thus counts events by event type.

[back to topic contents]

Worked example 3

This example shows how to tabulate event counts by event and another characteristic (integer age).

Ensure that lifecycle attributes are enabled by the following statement in model code:

```
options lifecycle_attributes = Person;
```

In `RiskPaths`, copy/paste the following code to `code/Tables.mpp`:

```
table snapshot untransformed Person X02_Events //EN Events by age
[ trigger_changes(lifecycle_counter) ]
{
   {
      unit //EN N
   } //EN Statistic
 * lifecycle_event //EN Event
 * self_scheduling_split(age,AGE_FERTILEYEARS) + //EN Age
};
```

Build and run the model, selecting the output table `X02_Events`.

The browser should display (with 7 of 28 age categories selected):

Event \ Age	(-∞,15)	[15,16)	[20,21)	[25,26)	[30,31)	[40,∞)	All
enter simulation	5,000	0	0	0	0	0	5,000
exit simulation (external)	0	0	0	0	0	0	0
DeathEvent	0	0	0	0	0	5,000	5,000
FirstPregEvent	0	80	584	209	63	0	4,633
Union1DissolutionEvent	0	0	4	5	0	63	160
Union1FormationEvent	0	127	413	128	28	0	3,495
Union2DissolutionEvent	0	0	0	0	0	55	59
Union2FormationEvent	0	0	4	2	3	72	141
UnionPeriod2Event	0	0	359	266	83	25	3,495

This table is like example 2 except for an additional classification dimension and a minor permutation of dimension order. The table shows the evolution of event frequencies over the life course. As expected, the margin over age has the same total event counts as example 2.

[back to topic contents]

Worked example 4

This example shows how to tabulate descriptive statistics for an attribute (`age`) at occurrences of a specific event (`FirstPregEvent`).

Ensure that lifecycle attributes are enabled by the following statement in model code:

```
options lifecycle_attributes = Person;
```

In `RiskPaths`, copy/paste the following code to `code/Tables.mpp`:

```
table snapshot untransformed Person X03_Events //EN Selected statistics for age at FirstPregEvent
[ trigger_changes(lifecycle_counter) && (lifecycle_event == LC_Person_FirstPregEvent) ]
{
  {
    unit, //EN N
    sum(age) / unit, //EN Mean
    minimum(age), //EN Min
    P5(age),  //EN P5
    P25(age), //EN Q1
    P50(age), //EN Median
    P75(age), //EN Q3
    P95(age), //EN P95
    maximum(age) //EN Max
  } //EN Statistic for age at FirstPregEvent
};
```

Build and run the model, selecting the output table `X03_Events`.

The browser should look like this, after moving expressions to rows and reducing display precision to two decimals:

The filter of this snapshot table restricts event snapshots to a specific kind of event, using the C++ `&&` operator. This same approach could be used to cross-tabulate a population at a specific event.

[back to topic contents]

Worked example 5

This example shows how lifecycle attributes can be used to output microdata when an event occurs.

Ensure that lifecycle attributes are enabled by the following statement in model code:

```
options lifecycle_attributes = Person;
```

In `RiskPaths`, copy/paste the following code:

```
options microdata_output = on;
entity Person
{
   hook write_microdata,
      trigger_entrances(lifecycle_event, LC_Person_Union1DissolutionEvent);
};
```

This code fragment writes a microdata record when the `Union1DissolutionEvent` occurs.

Build the model. In the run model screen, request microdata attributes `age` and `case_id`. Run the model and open the `Person` microdata tab. In the drop-down for size, select `All`.

The browser should display:

The screenshot shows `160` Person keys, which is the number of microdata records. This is the same as the number of `Union1DissolutionEvent` events shown in example 2.

Microdata can be output by hooking `write_microdata()` directly to an event implement function. The approach illustrated in this example can nevertheless be useful if the event implement function has multiple function return paths, or does not contain the required function call to implement function hooks. This approach also guarantees that `write_microdata()` is called only after all model code associated with the event has completed, which might include other entity functions hooked to the event implement function.

The technique of hooking `write_microdata` to `trigger_entrances` does not work for an event which causes the entity to exit the simulation. That's

because once the entity has exited the simulation no further events occur in it, including the internal self-scheduling event to which `write_microdata` is hooked.

[back to topic contents]

Worked example 6

This example shows how lifecycle attributes can help provide context in microdata output.

Ensure that lifecycle attributes are enabled by the following statement in model code:

```
options lifecycle_attributes = Person;
```

In `RiskPaths`, copy/paste the following code (replacing code in the previous example):

```
options microdata_output = on;
options microdata_write_on_exit = on;
entity Person
{
   hook write_microdata,
      trigger_changes(lifecycle_counter);
};
```

This code fragment writes a microdata record on simulation enter and on every event,
and when a `Person` exits the simulation. It's purpose in this example is just to produce some arbitrary microdata.

Build the model. In the run model screen, request microdata attributes `age` and `lifecycle_event`. Run the model and open the `Person` microdata tab. Drag `Person keys` so that it comes first in the row categorical list on the left, just before `lifecycle_event` (see screenshot below). Putting `Person keys` first sorts the microdata records by entity and timeline order.

The browser should display:

The column `lifecycle_event` in the display provides context for other microdata attributes (in this example, the `age` attribute).

The next example continues directly from this one, with no need to modify, rebuild, or rerun `RiskPaths`.

[back to topic contents]

Worked example 7

This example shows how lifecycle attributes can be used to filter microdata output in the UI.

It continues directly from the preceding example.

In the UI microdata viewer, click the `x` in the `Lifecycle event` drop-down to deselect all events, then open the drop-down and click `Union1FormationEvent` and `Union2FormationEvent` to select just them. Increase the `Size` to 100.

The browser should display:

Person keys	Lifecycle event	Person	Age
10002	Union1FormationEvent		24.26099921153568
20002	Union1FormationEvent		22.052372627648822
30002	Union1FormationEvent		17.05011124330298
60002	Union1FormationEvent		22.502915072767017
70002	Union1FormationEvent		17.845709212520806
80002	Union1FormationEvent		21.452578862824552
90002	Union1FormationEvent		20.991901608833572
100002	Union1FormationEvent		28.30145575751083
110002	Union1FormationEvent		24.680212872843498
110005	Union2FormationEvent		85.74865411176762
120002	Union1FormationEvent		18.10689608657391
130002	Union1FormationEvent		27.017285720574122

The microdata viewer displays age for all union formation events, in entity and timeline order.

[back to topic contents]

Local Random Streams

Model code can optionally specify that the state of random number streams be maintained locally for each entity rather than shared among entities. This can significantly reduce variance in run comparisons of models which simulate multiple instances of entities together, such as time-based models or case-based models with multiple entities in a case. It can also make run comparisons at the microdata level feasible for such models. Local random streams are not relevant for case-based models with a single entity per case. To jump to charts comparing the effect with and without local random streams in a highly interacting time-based model, click here.

Related topics

- Model Code
- Microdata Output
- Model Resource Use

Topic contents

- Background and overview
- Syntax and use
- Illustrative example Shows decoherence reduction graphically in a time-based model

Background and overview

Model code can draw random values from selected statistical distributions using built-in random number generator (RNG) functions, for example:

```
double x = RandUniform(1);
double y = RandNormal(2);
double z = RandPoisson(3);
```

These functions return pseudo-random streams of numbers. The streams appear random but are actually produced by a deterministic algorithm which generates a fixed sequence of values. That algorithm knows which value to return next by maintaining an internal state which changes from one function call to the next.

The sequence of numbers returned depends on the `SimulationSeed` for the run, on the run member (aka sub, replicate), on the case for case-based models, and on the *random stream number* (the small integer argument in RNG function calls).

The random stream number in an RNG function call specifies a distinct underlying random stream which produces values independent of those produced by other random streams. This avoids spurious interactions among unrelated random processes in the model. For example, values returned by calling `RandUniform(4)` in a `Fertility` module will not affect values returned by calling `RandUniform(6)` in a `Migration` module.

Independent random streams can reduce statistical noise in the difference of two model runs, reducing the run size needed to obtain reliable results for run differences. They also make microdata comparisons of two runs correspond better with model logic. For example, if there is no logical dependence between `Fertility` and `Migration` in the model, changing a `Fertility` parameter should not, logically, affect `Migration`. Had the same random stream, e.g. stream 4 in `RandUniform(4)`, been used in both `Fertility` and `Migration`, a call to `RandUniform(4)` in `Fertility` would affect the value returned in a subsequent call to `RandUniform(4)` in `Migration`. That would produce a spurious (but statistically neutral) interaction between `Fertility` and `Migration`. That can be avoided by using a different random stream in `Migration`, e.g. by calling `RandUniform(6)` to specify stream 6 instead of stream 4. Spurious correlation of random streams can be avoided by specifying a distinct random stream in each call to an RNG function throughout model code.

However, a model which simulates multiple instances of an entity together, e.g. multiple `Person` entities, could have spurious interactions of random streams among those entities. For example, a call to `RandUniform(4)` in `Fertility` in `Person` A will affect the result from a subsequent call in `Fertility` to `RandUniform(4)` in `Person` B, because the same random stream 4 is used in both. In a time-based model with many entities, a spurious interaction could extend from one entity to the entire population. Such spurious interactions do not affect the statistical validity of aggregate model results, but they can create additional statistical noise in run comparisons, and produce differences at the microdata level which are not explained by model logic.

This issue can be resolved by maintaining independent *local random streams* in each entity, rather than *global random streams* shared among entities. For example, using local random streams, a call to `RandUniform(4)` in `Person` A uses a different random stream from a call to `RandUniform(4)` in `Person` B.

Local random streams require additional memory in each entity to maintain the state of the pseudo-random number generator for each stream. This additional memory can be significant for time-based models with many entities and many random streams.

Local random streams must also be initialized distinctly in each entity so that different entities produce different random streams. That requirement is met by providing a unique key for each entity. The entity key is used to initialize local random streams independently in each each entity before it enters the simulation. The entity key needs to be stable from one run to another so that local random streams are the same for the same entity in two different runs. The code to specify the unique entity key is, in general, model dependent.

Given these trades, local random streams are not implemented by default in OpenM++.

However, a model coded to use local random streams can turn them off by changing a single line of model code and rebuilding, and reduce memory requirements when local random streams are not required for analysis.

[back to topic contents]

Syntax and use

Sections:

- Activation
- Entity key
- RNG use in entity initialization
- RNG use before simulation
- Memory use
- Internals

[back to topic contents]

Activation

Use the option `local_random_streams` to implement local random streams for all instances of a given entity, e.g.

```
options local_random_streams = Host;
```

to implement local random streams for all instances of the `Host` entity. Use multiple `options` statements to implement local random streams for more than one kind of entity.

During model build, a log message like

```
Entity 'Host' has 11 local random streams, of which 1 are Normal
```

is issued for each entity with local random streams. The message indicates how many of the local streams use a Normal distribution, because those use additional memory in the entity to maintain state.

[back to syntax and use]
[back to topic contents]

Entity key

As mentioned in Background and overview, an entity with local random streams needs a unique identifier to produce random streams which differ among entities.

The entity key needs to be not only unique in a run, but must also uniquely identify the same entity in the model runs being compared.

The entity key is provided by the entity member function `get_entity_key()`. `get_entity_key` returns a 64-bit value which is used internally by the built-in function `initialize_local_random_streams()` to initialize the local random streams of the entity before it enters the simulation.

If model code does not supply a definition for `get_entity_key()` a definition like the following is generated by the OpenM++ compiler:

```
uint64_t Host::get_entity_key()
{
   // This function definition was generated by omc because none was supplied in model code.
   // It returns the value of the built-in attribute entity_id.
   uint64_t entity_key = entity_id;
   return entity_key;
}
```

This default definition uses the built-in attribute `entity_id` as the entity key. `entity_id` is guaranteed to be unique for all entities in a run, making it a natural candidate for the entity key. However, `entity_id` might not identify the same entity in two different runs. That can happen if model code generates entities dynamically, or if the runs being compared have different numbers of entities or members (aka subs, replicates).

A model might already have an attribute which uniquely identifies the same entity in two different runs. For example, the model might be based on microdata which contains a unique personal identifier for each record/entity.

If the definition of `get_entity_key()` is supplied in model code, it would typically create the entity key using the values of one or more entity attributes before the entity enters the simulation.

The following hypothetical definition of `get_entity_key()` uses the helper function `xz_crc64` to combine the starting value of `age` and `time` to create the entity key. `xz_crc64` creates a 64-bit key using the `crc-64` open source checksum (hash) algorithm, and can use one value or combine multiple values together using successive calls.

```
uint64_t Host::get_entity_key()
{
   uint64_t key64 = 0;
   key64 = xz_crc64((uint8_t*)&age, sizeof(age), key64);
   key64 = xz_crc64((uint8_t*)&time, sizeof(time), key64);
   return key64;
}
```

[back to syntax and use]
[back to topic contents]

RNG use in entity initialization

This section applies only to calls to RNG functions in entity context. See section RNG use before simulation for information about using RNG functions outside of entity context before the simulation starts.

If an entity uses local random streams they must be initialized before use. If they are not initialized a fatal error like the following will be raised when the model is run:

```
Simulation error: RandUniform called with uninitialized local random streams.
```

By default, initialization is performed automatically when an entity enters the simulation, after starting values of all attributes have been assigned.

However, if model code in entity scope calls an RNG function before the entity enters the simulation, the local random streams of the entity will not have been initialized, causing the run-time error.

For example, the following code

```
void Host::Start()
{
   // Initialize all attributes (OpenM++).
   initialize_attributes();

   z_score = RandNormal(11);

   // Have the entity enter the simulation (OpenM++).
   enter_simulation();
}
```

assigns a random value to the `Host` attribute `z_score` before the `Host` enters the simulation. If `Host` uses local random streams, the run-time error would occur.

However, model code can explicitly initialize local random streams before the entity enters the simulation by calling the provided function `initialize_local_random_streams()`.

The following modified code

```
void Host::Start()
{
   // Initialize all attributes (OpenM++).
   initialize_attributes();

   // Need to explicitly initialize local random streams here
   // because they are used before the entity enters the simulation.
   initialize_local_random_streams();

   z_score = RandNormal(11);

   // Have the entity enter the simulation (OpenM++).
   enter_simulation();
}
```

initializes local random streams for the `Host` before the call to `RandNormal`, avoiding the error.

`initialize_local_random_streams()` uses `get_entity_key()` internally. If the definition of `get_entity_key()` is supplied in model code, assign all attributes used in `get_entity_key()` *before* the call to `initialize_local_random_streams()`.

`initialize_local_random_streams()` can be called (with no effect) even if the entity does not use local random streams, so there is no need to remove the call from model code if local random streams are not used in the entity.

Modgen specific: Modgen does not recognize the function `initialize_local_random_streams`. That causes an error when building the Modgen version of a x-compatible model. For x-compatible models, use code like:

```
#if !defined(MODGEN)
   initialize_local_random_streams();
#endif
```

[back to syntax and use]
[back to topic contents]

RNG use before simulation

Use of local random streams in one or more entities does not preclude use of global (shared) random streams.

Global random streams work normally in an entity not specified in a `local_random_streams` options statement.

Global streams also work normally in `PreSimulation` and in `Simulation`, before the simulation starts. For example, the `IDMM` model used in the Illustrative example uses two shared RNG streams in `Simulation`, one (random stream 5) to randomly infect a portion of the population at the beginning of the simulation:

```
// Create the initial population of Hosts, and infect some probabilistically.
for ( int nJ = 0; nJ < NumberOfHosts; nJ++ ) {
   auto prHost = new Host();
   prHost->Start();
   if (InitialDiseasePrevalence > RandUniform(5)) {
      prHost->Infect();
   }
}
```

and another (random stream 3) to construct the starting social network:

```
// Construct the initial social network.
int nHosts = asAllHosts->Count();
for ( int nJ = 0; nJ < nHosts; nJ++ ) {
   auto prHost = asAllHosts->Item( nJ );
   for (int nK = 0; nK < ContactsOutPerHost; nK++ ) {
      // Choose a host randomly from all hosts
      auto prContact = asAllHosts->GetRandom( RandUniform(3) );
      // Add it to the outgoing contacts link.
      // Note that if the contact happens to already be a contact, it will not be added.
      // because links contain no duplicates.
      if (prContact != prHost) {
         // do not link to self
         prHost->mlContactsOut->Add( prContact );
      }
   }
}
```

A global random stream does not affect a local random stream with the same number. However, best practice uses a unique random stream for each call to an RNG function in model code.

[back to syntax and use]
[back to topic contents]

Memory use

Memory impacts of local random streams are incorporated into the Resource Use Report in the run log, if resource use monitoring is on. To turn resource use monitoring on, add the following statement to model code:

```
options resource_use = on;
```

For example, here's the Resource Use Summary table from a 1,000,000 `Hosts` run of `IDMM` with local random streams:

```
+---------------------------+
| Resource Use Summary      |
+-------------------+-------+
| Category          |    MB |
+-------------------+-------+
| Entities          |   448 |
|   Host            |   448 |
|   Ticker          |     0 |
| Multilinks        |    63 |
| Events            |    95 |
| Sets              |    48 |
| Tables            |     0 |
+-------------------+-------+
| All               |   655 |
+-------------------+-------+
```

The same table without local random streams looks like this:

```
+---------------------------+
| Resource Use Summary      |
+-------------------+-------+
| Category          |    MB |
+-------------------+-------+
| Entities          |   384 |
|   Host            |   384 |
|   Ticker          |     0 |
| Multilinks        |    63 |
| Events            |    95 |
| Sets              |    48 |
| Tables            |     0 |
+-------------------+-------+
| All               |   591 |
+-------------------+-------+
```

In this example, memory requirements for the run increased from 591 MB to 655 MB in the version built with local random streams.

Additional lines appear in the entity member detail section of the report in a model with local random streams.

An additional row for the random state arrays is present in the member count table. The count is 1 if the entity does not use `RandNormal` or 3 if it does. This line counts the number of arrays used to maintain the state of local random streams, not the size of those arrays.

```
+----------------------------+
|     Host Members           |
+---------------------+-------+
| Member              | Count |
+---------------------+-------+
| Attributes          |    19 |
|   Built-in          |     6 |
|   Simple            |    12 |
|   Maintained        |     1 |
|   Link              |     0 |
| Events              |     3 |
| Increments          |     2 |
| Multilink           |     2 |
| Internal            |     6 |
| Array               |     0 |
| Foreign             |     0 |
| Random state arrays |     3 |
+---------------------+-------+
| All                 |    35 |
+---------------------+-------+
```

Additional lines appear in the entity member detail table.

Here's an extract from `IDMM` with local random streams:

```
+-----------------------------------------------------+
|     Host Members (detail)                           |
+---------------------------------------------+-------+
| member                                      | bytes |
+---------------------------------------------+-------+
| Attributes:                                 |       |
|   Built-in:                                 |       |
|     age                                     |     8 |
|     case_seed                               |     8 |
...
| Internal:                                   |       |
|     X01_History (in) om_duration            |     8 |
|     X02_Host_Events (inevent) event_count |     4 |
|     asAllHosts (current cell)               |     4 |
|     event_count (lagged)                    |     4 |
|     event_count (counter at lagged)         |     8 |
|     om_local_rng_streams_initialized        |     1 |
| Array:                                      |       |
| Foreign:                                    |       |
| Random state arrays:                        |       |
|     om_stream_X                             |    40 |
|     om_other_normal                         |     8 |
|     om_other_normal_valid                   |     1 |
+---------------------------------------------+-------+
| Sum of member bytes                         |   372 |
| Bytes per entity                            |   448 |
| Storage efficiency (%)                      |  83.0 |
+---------------------------------------------+-------+
```

An additional `Internal` member `om_local_rng_streams_initialized` is used to manage the entity life cycle of local random streams. Additional lines towards the end of the table show the memory used by the arrays which maintain random state.

[back to syntax and use]
[back to topic contents]

Internals

Local random streams are implemented using a multiplicative congruential generator (MCG), with an unused multiplier mentioned in the module `OM_ROOT/use/random/random_lcg200.ompp`. The multiplier of the MCG is 1187848453 and the modulus is the Mersenne prime 2^31-1.

For memory efficiency, the state of local random streams is maintained only for streams used in the entity. For example, `IDMM` model code in the Illustrative example uses 12 random streams, but only 10 are used in the `Host` entity. Local random state is maintained in each `Host` for just the 10 streams used in `Host`. The other 2 random streams are global and are used in `Simulation` to create the starting population before the simulation starts, as described in RNG use before simulation.

A random stream associated with the `RandNormal` RNG function requires additional memory to maintain state, because the underlying algorithm produces a pair of draws from N(0,1), and memory is required to save the second value of the pair which is returned by the next call to `RandNormal` for the stream.

Local random streams are seeded before the entity enters the simulation using 4 integer values: 1) the value returned by `get_entity_key()`, 2) the run seed for a time-based model or the case seed for a case-based model, 3) the run member (aka sub, replicate), and 4) the random stream number.

A local random stream thus depends on all 4 of these values, as required by the role of each.

These 4 values are combined using the `xz_crc64` hash function and the 64-bit result reduced to the allowed range of the MCG using the remainder of an appropriate integer division.

For more details, see the commented function definition of `initialize_local_random_streams` in the C++ file `src/om_declarations.h` which is generated by the OpenM++ compiler.

[back to syntax and use]
[back to topic contents]

Illustrative example

This example illustrates how local random streams can improve run coherence in a time-based model. Click here to jump directly to the graphical comparison.

Sections:

- Summary
- IDMM overview
- `Base` run
- `Variant` run
- Base-Variant coherence
- IDMM differences

[back to topic contents]

Summary

This example illustrates the effect of local random streams vs. global random streams on run coherence. It uses the time-based `IDMM` model with minor modifications. Microdata is output at 100 time points during the simulation, and later merged and compared between `Base` and `Variant` runs to measure how run coherence evolves during the simulation.

Four runs are used in this example , a `Base` and a `Variant` with two versions of the model:

1. `Base` run with shared random streams
2. `Variant` run with shared random streams
3. `Base` run with local random streams
4. `Variant` run with local random streams

The same version of `IDMM` was used for runs 1 and 2. A different version of `IDMM` was used for runs 3 and 4, the only difference being activation of local random streams.

All 4 runs have the same number of `Hosts` and an identical contact network created before the simulation starts using global random streams. That means that `entity_id` refers to the same `Host` in all 4 runs, so the default version of `get_entity_key()` can be used.

The `Base` runs have identical inputs in runs 1 and 3. The `Variant` runs have identical inputs in runs 2 and 4. A single parameter differs slightly between the `Variant` runs and the `Base` runs. That change causes only 2 entities to differ at the start of the `Variant` runs compared to the `Base` runs.

Aggregate outputs over time for the 4 runs are so similar that they are visually indistinguishable (see below). However, the degree of coherence at the microdata level between runs 3 and 4 (with local random streams) is noticeably different than that between runs 1 and 2 (with shared random streams).

Run-time resource reports were used to compare memory use for the two versions of `IDMM`.
Without local random streams, each `Host` entity was 384 bytes in size.
With local random streams, each `Host` entity was 448 bytes in size, an increase of 64 bytes or 17%.

[back to illustrative example]
[back to topic contents]

IDMM overview

`IDMM` simulates an interacting dynamic contact network of `Host` entities, together with a disease which can be transmitted over that contact network. The contact network is initialized randomly at the start of the simulation. During the simulation, each `Host` interacts with other `Hosts` in contact events. Each `Host` can change its connected `Hosts` in contact change events. Optionally, a `Host` can change a connected `Host` in a contact event, if that host is infectious.

During a contact event, the disease can propagate between the two `Hosts`, depending on the disease status of each `Host`. An infected `Host` progresses through 4 fixed length disease phases: susceptible, latent, infectious, and immune. On infection, a `Host` enters the latent phase, during which it is both asymptomatic and non-infectious. After the latent phase, the `Host` enters an infectious phase during which it can infect another `Host` during a contact event. After the infectious phase, the `Host` enters an immune phase. After the immune phase, the `Host` returns to the susceptible state.

Before the simulation starts, all `Host` entities are in the susceptible state. After a `Host` enters the simulation but before the first simulated event, it

is randomly infected at a given probability.

For this example, some mechanical changes were made to the version of `IDMM` in the OpenM++ distribution.

[back to illustrative example]
[back to topic contents]

`Base` run

The `Base` run consists of 5,000 `Hosts` simulated for 100 time units, with the initial probability of infection set to 0.1000. All other parameters are left at `Default` values, which are as follows:

```
parameters {
  int    NumberOfHosts              = 5000;  //EN Number of hosts
  int    ContactsOutPerHost         = 4;     //EN Number of outgoing contacts per host
  double MeanContactInterval        = 1.00;  //EN Mean time between social interactions
  double MeanChangeContactsInterval = 5.00;  //EN Mean time between changing social contact network
  double DumpSickContactProbability = 0.0;   //EN Probability of dumping a sick contact
  double InitialDiseasePrevalence   = 0.1000; //EN Disease prevalence at start of simulation
  double TransmissionEfficiency     = 0.10;  //EN Probability of transmitting disease during a social contact
  double LatentPhaseDuration        = 5.0;   //EN Duration of latent phase
  double ContagiousPhaseDuration    = 10.0;  //EN Duration of contagious phase
  double ImmunePhaseDuration        = 20.0;  //EN Duration of immune phase
  logical EnableChangeContactsEvent = TRUE;  //EN Enable the change contacts event
};
```

The `ini` file for the `Base` run looks like this:

```
[OpenM]
RunName = Base

[Parameter]
InitialDiseasePrevalence = 0.1000

[Microdata]
ToDb = yes
Host = report_time, disease_phase, age_infected
```

In the `Base` run, 501 `Hosts` are infected at the beginning of the simulation.

The time evolution of the susceptible population in Run 1 (`Base` with shared random streams) looks like this:

Base susceptibles, with shared random streams

Before the simulation starts, all 5,000 `Hosts` are in the susceptible state. Next, 501 `Hosts` are randomly infected and change disease status from susceptible to latent. This is the cause of the immediate drop of the count of susceptible `Hosts` from 5,000 to 4,499 in the chart. The following

plateau is caused by the length 5.0 latent period of the 501 infected hosts during which no forward infections occur. At the end of the plateau, the 501 Hosts leave the latent phase and enter the infectious phase. Then the infection spreads through the social network, progressively reducing the susceptible population. At around time 30.0, almost all Hosts have been infected and less than 25 susceptible Hosts remain. At time 35.0 the 501 initially infected Hosts lose protective immunity and become susceptible once again, causing a new plateau in the count of susceptible Hosts. Starting at time 41.0 progressively more Hosts lose protective immunity. The disease then propagates among the new and growing population of susceptible Hosts. The number of susceptible Hosts peaks at around time 57, and decreases subsequently as the disease infects more and more susceptibles. At time 82 less than 100 susceptible hosts remain. A new epidemic wave, with initial conditions smoothed out, commences towards the end of the simulation.

The same chart for Run 3 (Base with local random streams) looks like this:

Base susceptibles, with local random streams

This Base run with local random streams looks very similar to the Base run with shared random streams immediately above, despite all RNG draws differing during the simulation phase of the two runs. This is because the size of the population and the number of RNG draws dominates the effects of individual idiosyncratic RNG draws in the two runs. Put another way, the two versions of IDMM are statistically equivalent.

Exactly the same 501 Hosts are infected in Run 1 and Run 3, despite Run 3 using the version of IDMM with local random streams. That's because the initial infections are implemented during the creation of the starting population, before the simulation begins. Calls to RNG functions before the simulation starts use the usual shared random streams, not local random streams, as explained in RNG use before simulation.

[back to illustrative example]
[back to topic contents]

Variant run

The Variant runs are the same as the Base runs, except for a very slightly higher probability of initial infection of 0.1001 versus 0.1000 in the Base runs.

The ini file for the Variant runs looks like this:

```
[OpenM]
RunName = Variant

[Parameter]
InitialDiseasePrevalence = 0.1001

[Microdata]
ToDb = yes
Host = report_time, disease_phase, age_infected
```

503 Hosts are infected at the beginning of the simulation in the Variant run. That's 2 more than in the Base run. In the Variant runs, 501 of the 503 infected Hosts are identical to those infected at the beginning of the Base runs. The same 2 additional Hosts are infected in Variant runs 2

and 4.

The time evolution of the susceptible population in Run 2 (Variant with shared random streams) looks like this:

Variant susceptibles, with shared random streams

The aggregate effect of 503 initially infected Hosts vs. 501 initially infected Hosts is not visually perceptible, as might be expected.

The time evolution of the susceptible population in Run 4 (Variant with local random streams) looks like this:

Variant susceptibles, with local random streams

Run 2 and Run 4 are indistinguishable visually, despite having entirely different RNG draws in the simulation phase. They are by construction statistically equivalent.

[back to illustrative example]
[back to topic contents]

Base-Variant coherence

Base-Variant coherence is measured by counting the number of Hosts which have an identical infection history in the Base and Variant runs. The infection history of each Host is summarized in a 64-bit hash which combines the exact (fractional) ages of all previous infections experienced by the Host. Base-Variant coherence is evaluated at each of 101 time points in the runs by merging Base and Variant microdata population snapshots which are output at each time point.

The time evolution of Base-Variant coherence with shared random streams (runs 1 and 2) looks like this:

Base-Variant population coherence with shared random streams

Base-Variant coherence from time 0 to time 5 is constant at 4,998. As explained previously, exactly 2 Hosts differ between Base and Variant at the beginning of the runs, and no forward infections occur during the latent phase.

Base-Variant coherence is lost precipitously when the infectious phase begins, because of shared random streams among all Hosts in the population.

Virtually no coherence remains by time 72.

The time evolution of Base-Variant coherence with local random streams (runs 3 and 4) looks like this:

Base-Variant population coherence with local random streams

As in the previous chart, Base-Variant coherence starts at 4,998, with exactly 2 `Hosts` differing at the beginning of the simulation. The same initial plateau is present, but is difficult to distinguish visually. Coherence remains very high but is gradually lost as the effects of the tiny difference in initial conditions propagate in a causative way through the contact network and affect progressively more of the population.

Coherence remains above 99% until time 56. Coherence then decreases more rapidly until time 75, perhaps because of surging infections during that interval. Coherence drops to 90% by the end of the simulation.

The two versions of `IDMM` in this example, the one with and the one without local random streams, are statistically equivalent at the aggregate level. However, the difference in coherence at the micro level is striking.

The population of `Hosts` in `IDMM` is highly connected and interactive. Models with less connected and less interactive populations may well have very high coherence if they use local random streams.

[back to illustrative example]
[back to topic contents]

IDMM differences

The version of `IDMM` used in this example differs slightly from the version in the OpenM++ distribution in `OM_ROOT/models/IDMM`.

The differences are as follows:

1. The `Host` attributes `age_infected` and `history_hash` were added and implemented to measure coherence between runs.
2. Code to infect a portion of the population at the beginning of the simulation was moved from `Host::Start` to the `Simulation` function, so that the same global random streams are used in all 4 runs in the example, ensuring that the starting populations are the same.
3. A custom version of `Host::get_microdata_key()` was added to provide a unique microdata key for each `Host` at each value of `report_time`.
4. `initialize_local_random_streams` was called explicitly in `Host::Start` to permit use of `RandUniform` later in `Host::Start` to schedule the first `ContactEvent` and the first `ChangeContactsEvent`.
5. `Ticker::TickEvent` was modified to write microdata for each `Host` at each time tick.
6. The `REPORT_TIME` range was enlarged from 20 to 100.
7. The options `local_random_streams`, `microdata_output`, and `resource_use` were enabled.
8. Two unused `Host` attributes `z_score` and `l_score` were added and implemented to test the local random streams implementation of `RandNormal` and `RandLogistic`.

[back to illustrative example]

[back to topic contents]

Memory Use

This topic opens with a general discussion of memory use in models and follows with a compendium of techniques to optimize memory use in time-based models.

Related topics

- Model Code
- Model Resource Use

Topic contents

- Introduction and Background
- Bag of Tricks Techniques to economize memory

Introduction and Background

This topic describes techniques to simulate large populations of entities in time-based models by reducing per-entity memory requirements. It is not applicable to case-based models which by design can simulate populations of unlimited size in a fixed amount of memory.

A computer's physical memory is a limited fixed resource, whether the computer is a desktop, a server, or a component of a cloud service. If memory requested by an application exceeds the computer's available physical memory, the request may be denied, causing the application to fail. More typically the computer will attempt to ration physical memory by swapping less-used portions of application memory to disk and bringing them back on demand, a process called paging. Paging is very slow relative to the speed of physical memory, but can work well if memory demands over time are concentrated in some regions of memory while being infrequent in others.

Models can request large amounts of memory, and use that memory frequently and intensively. A time-based model with a large interacting population of entities will access and update those entities frequently during a run over large regions of memory. Because the model accesses large regions of memory in a scattered rather than a concentrated way, models respond poorly to paging. When models are starved for memory and start paging, they may slow down by orders of magnitude and become unusable. So in practice the entities in the population need to all fit into the physical memory of the target computer.

Reducing per-entity memory use increases the maximum number of entities which can be in memory simultaneously. The techniques in this topic can help reduce per-entity memory use without changing model logic.

Time-based models have an inherent tradeoff between population size and model complexity, because the size of an entity increases with model complexity. Case-based models have no such tradeoff between population size and complexity, but they can't represent large interacting populations. When the modelling problem to be solved requires both large population size and high complexity, it may be possible to factor it into two models which are run sequentially: an upstream time-based model which simulates an interacting population with limited complexity paired with a downstream case-based model with no population interactions but unlimited complexity. An example is the combination HPVMM and OncoSim, where the upstream time-based HPVMM model simulates infectious disease dynamics and the downstream case-based OncoSim model simulates disease screening, diagnosis, treatment, health consequences, and health system resources and costs. In the HPVMM-OncoSim pairing, HPVMM results on disease incidence (rates of infection) are communicated as inputs to the downstream OncoSim model.

[back to topic contents]

Bag of Tricks

This subtopic contains links to sections which describe techniques to manage memory use. It is not always appropriate to apply one of these techniques. It may not be worth the effort or additional model complexity, or there may be a functionality trade-off which is not worth making. The Entity Member Packing report can help identify which techniques will be most fruitful. The list of links is followed by another list of BoT candidates.

- Exploit the resource use report Use the report to focus efforts fruitfully
- Suppress table groups: Use and suppress table groups
- Change time type to float: Use `time_type` to halve time storage
- Use value_out with flash tables: Use `value_out` instead of `value_in`, especially with flash tables
- Enable entity packing: Enable the `entity_member_packing` option

- Use mutable real type: Use `real` for floating point values
- Prefer range and classification to int: They take fewer bytes of storage
- Use bitset instead of bool array: Store large arrays of `bool` efficiently
- Purge available entity list: Purge the available list after all entities of a given type are gone

BoT candidates:

- compute rather than store
- use smaller C types
- hook to self-scheduling events, e.g. self_scheduling_int(age)
- be economical with events
- be economical with tables
- avoid ordinal statistics
- use a unitary Ticker actor to push common characteristics to the population, e.g. year

[back to topic contents]

Exploit the resource use report

Software developers often guess wrong about the causes of high resource use. It is best to gather data before embarking on efforts to improve efficiency. The Model Resource Use report was designed for that purpose.

[back to BoT]
[back to topic contents]

Suppress table groups

Organize tables into functional groups using `table_group` statements. Then, use `retain_tables` to keep only the tables needed for the current purpose when the model is built. When a table is suppressed when a model is built, memory savings accure both from the memory for the table cells and for entity members associated with the table.

Organizing tables into groups and retaining only those required for immediate needs allows a model to have many tables without paying a high memory cost. If a diagnostic table is required after exploring run results, a variant of the model can be built with that table retained and the simulation re-run.

[back to BoT]
[back to topic contents]

Change time type to float

The `Time` type of a model can be changed from the default `double` to `float` by inserting the following statement in model code:

```
time_type float;
```

The `Time` type is ubiquitous in models. It is used in attributes, events, and internal entity members. By default, `Time` wraps the C++ type `double`, which is a double-precision floating point number stored in 8 bytes. The `time_type` statement allows changing the wrapped type to `float`, which is stored in 4 bytes. This can reduce memory use. For example, here is the summary report for the 1 million GMM run used to illustrate the Model Resource Use topic where `time_type` is `double` (the default value):

```
+---------------------------+
| Resource Use Summary      |
+-------------------+-------+
| Category          |    MB |
+-------------------+-------+
| Entities          |  1924 |
|   Doppelganger    |   552 |
|   Person          |  1372 |
|   Ticker          |     0 |
| Multilinks        |    10 |
| Events            |    80 |
| Sets              |   196 |
| Tables            |     0 |
+-------------------+-------+
| All               |  2213 |
+-------------------+-------+
```

Here is the report for the same 1 million run with `time_type` set to `float`:

```
+---------------------------+
| Resource Use Summary      |
+-------------------+-------+
| Category          |    MB |
+-------------------+-------+
| Entities          |  1566 |
|   Doppelganger    |   515 |
|   Person          |  1050 |
|   Ticker          |     0 |
| Multilinks        |    10 |
| Events            |    80 |
| Sets              |   196 |
| Tables            |     0 |
+-------------------+-------+
| All               |  1854 |
+-------------------+-------+
```

In this example, memory usage of the `Person` entity was 23% less with `time_type` set to `float` compared to `double`.

A `float` has a precision of about 7 decimal digits. Roughly, that means that the `float` time value 2025.123 is distinguishable from 2025.124, which is a precision of ~8 hours. A model with a time origin of 0 and maximum time of 100 years would have higher `float` precision than that. The run-time function `GetMinimumTimeIncrement()` gives the actual precision of `Time` values in a model. The actual precision is based on `time_type` and the maximum `Time` value required by the model in a previous call to `SetMaxTime()` in model code.

Changing `time_type` to `float` may affect model results due to the reduced precision of `Time` values. If model logic is well represented by `float` precision, such differences are likely to be statistical. That can be verified by comparing model results with `time_type float` vs. `time_type double` by changing a single line of model code.

[back to BoT]
[back to topic contents]

Use value_out with flash tables

Flash tables are entity tables which tabulate at instantaneous points in time. They do that using an attribute like `trigger_changes(year)` in the table filter which becomes instantaneously true and then immediately false in a subsequent synthetic event. Because an increment to a flash table is instantaneous it has identical 'in' and 'out' values. That means that a flash table using 'value_in' will produce the same values as 'value_out'. However, `value_in` in a table causes the compiler to create an additional member in the entity to hold the 'in' value of an increment. For flash tables, this memory cost can be avoided by using 'value_out' instead of 'value_in'.

[back to BoT]
[back to topic contents]

Enable entity packing

Members of entities can be packed more efficiently by turning on the `entity_member_packing` option, but there is a trade-off. For mode information see Entity Member Packing.

[back to BoT]
[back to topic contents]

Use mutable real type

Floating point values can be declared in model code using the `real` type. By default, `real` is the same as the C++ type `double`, but it can be

changed to the C++ type `float` by inserting the following statement in model code:

```
real_type float;
```

This single statement will change all uses of `real` from `double` to `float`, which will halve the storage requirements of `real' values.

A `float` has a precision of around 7 digits, so can represent a dollar amount of 12345.67 to an accuracy of 1 cent.

Because a single `real_type` statement changes the meaning of `real` throughout, it is easy to assess to what extent changing `real` from `double` to `float` affects results. This provides more flexibility than changing (say) `double` to `float` in code.

[back to BoT]
[back to topic contents]

Prefer range and classification to int

Values of type Range or Classification are automatically stored in the smallest C type which can represent all valid values. This can reduce memory use. For example, if `YEAR` is declared as

```
range YEAR //EN Year
{
   0, 200
};
```

a member `year`

```
entity Person {
   YEAR year;
};
```

declared with type `YEAR` will be stored efficiently in a single byte. In contrast, of `year` was declared as `int` it would require 4 bytes.

[back to BoT]
[back to topic contents]

Use bitset instead of bool array

The `bool` type takes one byte of storage, even though a byte contains 8 bits. Some models use large arrays of `bool` in entity members, e.g.

```
entity Person {
   bool was_primary_caregiver[56][12];
}
```

which records whether a `Person` was a primary caregiver in each month of 56 possible working years during the lifetime. The Model Resource Use report would show that the `was_primary_caregiver` array member of `Person` consumes 672 bytes of memory in each `Person`, a significant amount for a time-based model with a large population.

The same information could be stored in a foreign member of `Person` using the a C++ `std::bitset`. A code sketch follows:

```
typedef std::bitset<56*12> ym_bools; // flattened bit array of 56 years and 12 months
...
entity Person {
   ym_bools was_primary_caregiver;
}

std::size_t ym_index(std::size_t year, std::size_t month) {
   return 12 * year + month;
}
```

Then model code like

```
ptCareGiver->was_primary_caregiver[nEarningIndex][nM] = true;
```

could be replaced by similar functionally equivalent code

```
ptCareGiver->was_primary_caregiver[ym_index(nEarningIndex,nM)] = true;
```

In this example, replacing the array of `bool` by a `std::bitset` reduces storage requirements from 672 bytes to 84 bytes, a significant saving for each `Person` entity.

If the `bool` array was 1-dimensional rather than 2-dimensional, the code would be simpler.

Possibly, a general wrapper class `bitset2D` could be added to OpenM++ runtime support to avoid changing model code at all, e.g.

```
#include "bitset2D.h"
...
typedef bitset2D<56,12> ym_bools; // 2-D bit array of 56 years and 12 months
...
entity Person {
    ym_bools was_primary_caregiver;
}
```

Then existing model code like

```
ptCareGiver->was_primary_caregiver[nEarningIndex][nM] = true;
```

would require no changes.

[back to BoT]
[back to topic contents]

Purge available entity list

Depending on the model design, an entity type might be used only at a particular phase of the simulation. For example, an `Observation` entity migh only be used during the creation of the starting population. OpenM++ maintains pools of entities which have exited the simulation which are available for potential reuse. If there will be no reuse of an entity type, the corresponding pool can be emptied and the memory reclaimed by a function call like

```
Observation::free_available();
```

[back to BoT]
[back to topic contents]

Microdata Output

Microdata output allows a model to output records containing the values of selected entity attributes during a run for later use. This topic describes microdata output from a model developer perspective.

Related topics

- Model Code
- Event Trace

Topic contents

- Introduction
- Topic outline
- Quick start How to build and run a model with microdata output
- Worked example 1a Entity life cycle
- Worked example 1b Entity life cycle with event context
- Worked example 1c Entity life cycle with event filtering
- Worked example 2a Output using a hook to a model event
- Worked example 2b Output using a hook to a self-scheduling attribute
- Worked example 2c Output by calling `write_microdata` in model code
- Worked example 3 Database output in a time-based model
- Worked example 4 Database output in a complex case-based model
- Microdata output modes Text mode and database mode
- Microdata output control Enabling microdata and microdata attributes
- Run-time settings Run-time settings
- Build-time settings Build-time settings
- Writing microdata from model code Controlling microdata output from model code
- The microdata key The purpose of the microdata key and how to set it

Introduction

A model built with microdata output capability can output records containing the values of entity attributes. As well as attribute values, each microdata output record contains a microdata key to match corresponding records between runs.

By default, a model does not have microdata output capability. See Enabling microdata output or Quick start on how to build a model with microdata output capability.

Two microdata output modes are supported: text mode and database mode. Text mode is targeted more to model developers, while database mode is targeted more to users of production models and to future OpenM++ run-time tabulation functionality. Both modes can be active in the same run.

Text mode writes microdata to text files in `csv` format. Text mode can filter output at run using event context. Text mode output can include an additional column showing the event context of the record.

Database mode writes microdata to the model database, from which it can be extracted using `dbcopy` or an API. Database mode will be used for future OpenM++ run-time tabulation functionality, including microdata comparisons between runs.

Microdata output is controlled by run-time settings, build-time settings, and model code.

Run-time settings specify which attributes are output during a run, provided the model was built with microdata output capability. All attributes are available for selection at run-time without rebuilding the model. Some run-time settings apply only to text mode. Those text mode settings can filter records by event context and can create an additional column showing the event context for each record.

Build-time settings are statements in model code which make the model capable of microdata output and control related warning messages. Build-time settings can also (optionally) determine when microdata output is written in the entity life cycle: on entrance, on exit, or on the occurrence of an event.

Model code can write microdata explicitly by calling the supplied entity member function `write_microdata`. The `write_microdata` function can be hooked to an existing entity function such as the implementation function of an event.

[back to topic contents]

Topic outline

Quick Start shows how to build a model capable of microdata output and how to activate that capability in a model run.

The quick start is followed by several worked examples with illustrative inputs and outputs, mostly using the `RiskPaths` model.

The first group of examples
entity life cycle,
entity life cycle with event context, and entity life cycle with event filtering
illustrate how to probe the life cycle of entities using microdata text mode.

A second group of examples
output using a hook to a model event,
output using a hook to a self-scheduling attribute, and
output by calling `write_microdata` in model code
illustrate how to control when microdata output occurs using model code.

The next example illustrates database output in a time-based model. The example outputs microdata for all entities in the `IDMM` model at the end of the run. A `Base` run and a `Variant` are performed, and the results compared at the microdata level using files exported by `sbcopy`.

The final example illustrates database output in a case-based model. The example outputs microdata for all entities in the complex case-based model `OncoSim`. A `Base` run and a `Variant` are performed, and summary microdata indicators (years lived and health system cost) are output for each Person at the end of each case. The results are exported by `dbcopy` and analyzed to identify all cases which differed due to the parameter change in the Variant run, and by how much.

The worked examples are followed by subtopics which explore specifics in more detail:

- Microdata output modes
- Enabling microdata output
- Run-time settings
- Build-time settings
- Writing microdata from model code
- The microdata key

[back to topic contents]

Quick start

This subtopic contains the following sections.

- 1. Build model with microdata output capability
- 2. Create model `ini` file with microdata output options
- 3. Run model using microdata output

[back to topic contents]

1. Build model with microdata output capability

Add the following statements to the model source code file `RiskPaths/code/ompp_framework.ompp`:

```
options microdata_output = on;
options microdata_write_on_exit = on;
```

Build the Release version of RiskPaths.
In Windows, the model executable will be `RiskPaths/ompp/bin/RiskPaths.exe`.
In Linux, the model executable will be `RiskPaths/ompp-linux/bin/RiskPaths`.

[back to quick start]
[back to topic contents]

2. Modify model `ini` file with microdata output options

In the same folder as the RiskPaths executable there may already be a copy of the default model `ini` file `RiskPaths.ini`. If not create it using your IDE or a text editor such as Notepad.

Edit `RiskPaths.ini` to have the following content:

```
[Parameter]
SimulationCases = 5

[Microdata]
ToCsv = yes
Person = age, union_status, parity_status
```

[back to quick start]
[back to topic contents]

3. Run model using microdata output

Launch the model in its `bin` directory using the `ini` file created in the previous step.

```
RiskPaths -ini RiskPaths.ini
```

In Windows you can run the Release version of RiskPaths from inside Visual Studio as follows:

- `Solution Configurations` to `Release` and `Solution Platforms` to `x64`
- `Project Properties > Configuration Properties > Debugging > Command Arguments` to `-ini RiskPaths.ini`
- `Project Properties > Configuration Properties > Debugging > Working Directory` to `$(TargetDir)`
- To launch the model, do `Debug > Start without debugging` or Press `Ctrl-F5`.

When the model run completes, the file `RiskPaths.Person.microdata.csv` should be present in the model `bin` directory and look like this:

```
key,age,union_status,parity_status
1,100,2,1
2,100,2,1
3,100,2,1
4,100,0,1
5,100,0,1
```

or formatted as a table, like this:

key	age	union_status	parity_status
1	100	2	1
2	100	2	1
3	100	2	1
4	100	0	1

key	age	union_status	parity_status
5	100	0	1

The run-time settings output the attributes `age`, `union_status`, and `parity_status`. The leading column `key` can be used to match microdata records between runs. The build-time option `microdata_write_on_exit` causes a microdata record to be written whenever an entity leaves the simulation. In `RiskPaths` there is no mortality and `Person` entities exit the simulation at age 100. The values of `union_status` and `parity_status` are those at that age, for each `Person` entity in the run.

The model log contains the following warning, which is expected.

```
Warning : model can expose microdata at run-time with output_microdata = on
```

[back to quick start]
[back to topic contents]

Worked example 1a

This example is the first of three which probe entity life cycle using microdata output in text mode. It continues the quick start example to output multiple microdata records for a single entity: when it enters the simulation, at each event, and when it leaves the simulation.

In `ompp_framework.ompp`, change the build-time microdata settings to

```
options microdata_output = on;

options microdata_write_on_enter = on;
options microdata_write_on_exit = on;
options microdata_write_on_event = on;
```

Change the run-time settings in `RiskPaths.ini` to consist of only one case

```
[Parameter]
SimulationCases = 1

[Microdata]
ToCsv = yes
Person = age, union_status, parity_status
```

and run the model.

Here's the resulting microdata output in `RiskPaths.Person.microdata.csv`, with some rows elided.

key	age	union_status	parity_status
1	0	0	0
1	1	0	0
1	2	0	0
1	3	0	0
...	...	...	...
1	22.5	0	0
1	23	0	0
1	24	0	0
1	24.2609992115357	1	0
1	25	1	0
1	25.2609992115357	1	0
1	26	1	0

key	age	union_status	parity_status
1	26.5378127283906	1	1
1	26.5378127283906	1	1
1	27	1	1
1	27.2609992115357	1	1
1	27.2609992115357	2	1
1	27.5	2	1
1	28	2	1
1	29	2	1
1	29.2609992115357	2	1
1	30	2	1
...	...	...	...
1	99	2	1
1	100	2	1
1	100	2	1
1	100	2	1

The microdata output shows the values of the attributes at every event in the life cycle. Multiple microdata records can occur at the same age due to multiple tied events at that age.

Worked example 1b

This example is the second of three which probe entity life cycle using microdata output in text mode. It continues the previous example, adding event context information to each microdata record.

Leave the build-time microdata settings in `ompp_framework.ompp` unchanged from the previous example:

```
options microdata_output = on;

options microdata_write_on_enter = on;
options microdata_write_on_exit = on;
options microdata_write_on_event = on;
```

Activate the `CsvEventColumn` option by modifying the run-time settings in `RiskPaths.ini` so that it looks like this:

```
[Parameter]
SimulationCases = 1

[Microdata]
ToCsv = yes
CsvEventColumn = true
Person = age, union_status, parity_status
```

Run the model.

Here's the resulting microdata output in `RiskPaths.Person.microdata.csv`, with some rows elided.

key	event	age	union_status	parity_status
1	(no event)	0	0	0
1	om_ss_event	1	0	0
1	om_ss_event	2	0	0

key	event	age	union_status	parity_status
1	om_ss_event	3	0	0
...	...	...	...	...
1	om_ss_event	22.5	0	0
1	om_ss_event	23	0	0
1	om_ss_event	24	0	0
1	Union1FormationEvent	24.2609992115357	1	0
1	om_ss_event	25	1	0
1	om_ss_event	25.2609992115357	1	0
1	om_ss_event	26	1	0
1	FirstPregEvent	26.5378127283906	1	1
1	om_ss_event	26.5378127283906	1	1
1	om_ss_event	27	1	1
1	om_ss_event	27.2609992115357	1	1
1	UnionPeriod2Event	27.2609992115357	2	1
1	om_ss_event	27.5	2	1
1	om_ss_event	28	2	1
1	om_ss_event	29	2	1
1	om_ss_event	29.2609992115357	2	1
1	om_ss_event	30	2	1
...	...	...	...	...
1	om_ss_event	99	2	1
1	om_ss_event	100	2	1
1	DeathEvent	100	2	1
1	DeathEvent	100	2	1

The microdata output now contains an `event` column showing the name of the event being implemented when each microdata record was output. There is no event at the beginning of a case in a case-based model like `RiskPaths`, so when the first entity in the case enters the simulation `(no event)` is shown in the `event` column. If the event associated with microdata output is a self-scheduling event, `om_ss_event` is shown in the `event` column. The internal self-scheduling event for an entity implements all self-scheduling attributes in the entity. Note that Event Trace can be used to obtain more information about events, including the names of self-scheduling events.

The final three microdata output records all occur at age 100. Here's a detailed explanation of each of these apparent duplicate records:

The first is from the self-scheduling event which maintains the derived attribute `self_scheduling_int(age)`. That derived attribute is in turn used in the declaration of the identity attribute `integer_age`:

```
actor Person                //EN Individual
{
   //EN Current integer age
   LIFE integer_age = COERCE( LIFE, self_scheduling_int(age) );
...
```

The second is from the event `DeathEvent` which is triggered by model logic and the `ProbMort` parameter immediately when `integer_age` is 100:

```
TIME Person::timeDeathEvent()
{
   TIME event_time = TIME_INFINITE;
   if (CanDie)
   {
      if (ProbMort[integer_age] >= 1)
      {
         event_time = WAIT(0);
      }
...
```

The third occurs when the entity leaves the simulation, because the option `microdata_write_on_exit` is `on` in the example. The event `DeathEvent` was the active event when the entity left the simulation, so that's what's shown in the `event` column.

Although it's not illustrated in this example, the name in the `event` column can be prefixed by a `*`. This indicates that the active event is in a different entity than the one being output. This can occur in a time-based model or in a case-based model with multiple entities in a case. For example a `ChildBirth` event in a Person entity could cause a new Person entity to enter the simulation and generate a microdata output record. The microdata record for the newborn would contain `*ChildBirth` in the `event` column to indicate that the active event was in a different entity than the microdata record.

[back to topic contents]

Worked example 1c

This example is the third of three which probe entity life cycle using microdata output in text mode. It extends the previous example by filtering on specific events.

Leave the build-time microdata settings in `ompp_framework.ompp` unchanged from the previous example:

```
options microdata_output = on;

options microdata_write_on_enter = on;
options microdata_write_on_exit = on;
options microdata_write_on_event = on;
```

Modify the run-time settings in `RiskPaths.ini` to increase the number of cases to 5000, and restrict output to two named events using the `Events` option:

```
[Parameter]
SimulationCases = 5000

[Microdata]
ToCsv = yes
CsvEventColumn = true
Person = age, union_status, parity_status
Events = Union1FormationEvent, FirstPregEvent
```

Run the model.

The resulting microdata output `RiskPaths.Person.microdata.csv` has 8,128 records and looks like this:

key	event	age	union_status	parity_status
1	Union1FormationEvent	24.2609992115357	1	0
1	FirstPregEvent	26.5378127283906	1	1
2	Union1FormationEvent	22.0523726276488	1	0
2	FirstPregEvent	24.6780778011483	1	1
3	Union1FormationEvent	17.050111243303	1	0
3	FirstPregEvent	20.024664717724	1	1
4	FirstPregEvent	17.4107170399441	0	1
5	FirstPregEvent	24.1577392012077	0	1
6	Union1FormationEvent	22.502915072767	1	0

key	event	age	union_status	parity_status
6	FirstPregEvent	24.7534475294375	1	1
...	...	...	...	...

This `csv` file can be used to perform multivariate statistical analysis. For example, the `csv` file can be opened in Excel, filtered to just `FirstPregEvent` and a histogram generated to visualize the first birth distribution by age:

The data could be additionally filtered in Excel using the `union_status` column to visualize how union status affects the age distribution to produce the overall pattern.

[back to topic contents]

Worked example 2a

This example is the first of three illustrating control of microdata output at build time using model code. It outputs microdata whenever a specific event occurs using a `hook` in model code, specifically whenever `FirstPregEvent` occurs in `RiskPaths`.

In `RiskPaths`, prepare the event implement function for hooks by adding the required statement at the end of the event implement function `FirstPregEvent`:

```
void Person::FirstPregEvent()
{
    parity_status = PS_PREGNANT;
    hook_FirstPregEvent();
}
```

Next, add code to hook the built-in function `write_microdata` to `FirstPregEvent`:

```
actor Person {
    hook write_microdata, FirstPregEvent;
};
```

In `ompp_framework.ompp`, turn off options which automatically write microdata, which were previously turned on in example 1.

```
//options microdata_write_on_enter = on;
//options microdata_write_on_exit = on;
//options microdata_write_on_event = on;
```

The statements inserted in example 1 were commented to revert to the default value `off`. This means that only explicit calls to `write_microdata` will generate microdata output.

Set the number of cases to 20 in `RiskPaths.ini`:

```
[Parameter]
SimulationCases = 20

[Microdata]
ToCsv = yes
Person = age, union_status, parity_status
```

Run the model.

The microdata output file `RiskPaths.Person.microdata.csv` should look like this:

key	age	union_status	parity_status
1	26.5378127283906	1	1
2	24.6780778011483	1	1
3	20.024664717724	1	1
4	17.4107170399441	0	1
5	24.1577392012077	0	1
6	24.7534475294375	1	1
7	18.2797585879836	1	1
8	22.110326319997	1	1
9	21.2430736420085	1	1
10	29.168835553187	1	1
12	37.7955780112222	2	1
14	26.9550960057145	1	1
15	21.6012847802494	0	1
16	20.3178392448776	1	1
18	22.8298415328563	1	1
19	26.7999269606788	1	1
20	19.0257883348614	1	1

The microdata file shows the values of attributes at all occurrences of the `FirstPregEvent` in the run. It could, for example, be used to chart the distribution of age at first birth using a downstream application like Excel or R, similar to example 1c.

[back to topic contents]

Worked example 2b

This example is the second of three illustrating control of microdata output at build time using model code. It outputs microdata records giving a snapshot of an entity at each integer age, using a `hook` to a self-scheduling attribute.

Change the hook in the previous example to

```
actor Person {
    hook write_microdata, self_scheduling_int(age);
};
```

and simulate a single case by modifying RiskPaths.ini:

```
[Parameter]
SimulationCases = 1

[Microdata]
ToCsv = yes
Person = age, union_status, parity_status
```

Run the model. Microdata output should look like this:

key	age	union_status	parity_status
1	1	0	0
1	2	0	0
1	3	0	0
1	4	0	0
...	...	...	...
1	26	1	0
1	27	1	1
1	28	2	1
1	29	2	1
...	...	...	...
1	100	2	1

The microdata output contains a snapshot of the attributes at each integer age.

The technique of hooking `write_microdata` to a self-scheduling or a trigger attribute will not output microdata if the current event causes the entity to exit the simulation. That's because once the entity has exited the simulation no further events occur in it, including the internal self-scheduling event to which `write_microdata` is hooked.

[back to topic contents]

Worked example 2c

This example is the third of three illustrating control of microdata output at build time using model code. It outputs microdata directly by calling the entity function `write_microdata` explicitly in model code.

Remove any changes to `RiskPaths` model code made in previous examples.

In `ompp_framework.ompp`, insert the single statement

```
options microdata_output = on;
```

Insert a call to `write_microdata` in the implementation function of the `FirstPreg` event in the module `Fertility.mpp`:

```
void Person::FirstPregEvent()
{
    parity_status = PS_PREGNANT;
    write_microdata();
}
```

Set the run-time settings in `RiskPaths.ini` as follows:

```
[Parameter]
SimulationCases = 5

[Microdata]
ToCsv = yes
Person = age, union_status, parity_status
```

Run the model.

Output should look as follows:

```
key,age,union_status,parity_status
1,26.5378127283906,1,1
2,24.6780778011483,1,1
3,20.024664717724,1,1
4,17.4107170399441,0,1
5,24.1577392012077,0,1
```

This example could be accomplished without using a direct call to `write_microdata`. In a more complex model, a call to `write_microdata` could be placed inside conditional model logic, for example to output microdata when a rare causative path is taken in model logic, to probe correctness.

[back to topic contents]

Worked example 3

This example outputs microdata in database mode for the time-based model `IDMM`. Two runs `Base` and `Variant` are performed with an incremental parameter change. Microdata with infection status is output for all `Host` entities at the end of the run. Each run consists of multiple replicates. The `dbcopy` utility is used to extract the microdata for the two runs. Excel is used to import the microdata and construct a table showing the concordance of disease state at the microdata level between the Base and Variant runs.

Modify the `IDMM` model to activate microdata output when entities leave the simulation by adding the following statements to `ompp_framework.ompp`:

```
options microdata_output = on;
options microdata_write_on_exit = on;
```

Rebuild the model.

Arrange that `IDMM` uses the file `IDMM.ini` to get run-time settings (see quick start), and set the contents of `IDMM.ini` to create a run named `Base` as follows:

```
[OpenM]
SubValues = 5
Threads = 5
RunName = Base

[Parameter]
NumberOfHosts = 10000
ImmunePhaseDuration = 20.0

[Microdata]
ToDb = yes
Host = disease_phase
```

These settings create a `Base` run with 5 replicates, each with a population of 10,000 `Host` entities.

Run the model.

The log file should contain a line like

```
2023-01-13 17:01:04.874 Warning : model can expose microdata at run-time with output_microdata = on
```

which indicates that the version of IDMM is capable of writing microdata. It should also contain a line similar to

```
2023-01-13 17:01:08.295 Writing microdata into database, run: 103
```

which indicates that the model is merging microdata from replicates into the database when the run completes.

Change the file `IDMM.ini`, modifying `RunName` and `ImmunePhaseDuration` for a second run named `Variant`:

```
[OpenM]
SubValues = 5
Threads = 5
RunName = Variant

[Parameter]
NumberOfHosts = 10000
ImmunePhaseDuration = 22.0

[Microdata]
ToDb = yes
Host = disease_phase
```

The `Variant` run is the same as the `Base` run, except for a 10% increase in the duration of protective immunity from a previous infection.

Run the model.

The model database now contains results for the two runs `Base` and `Variant`.

Open a command shell. Change the current directory to the ompp/bin directory of the IDMM model. Run `dbcopy` to extract the microdata results from the model database to `csv` files using the command

```
dbcopy -dbcopy.To csv -dbcopy.ModelName IDMM
```

By default `dbcopy` looks for a model database in the current directory, so it's not necessary in this example to provide it the path of the model database.

Console output should be similar to the following:

```
C:\Development\X\ompp\models\IDMM\ompp\bin>%OM_ROOT%\bin\dbcopy -dbcopy.To csv -dbcopy.ModelName IDMM
2023-01-13 17:01:45.580 Model IDMM
2023-01-13 17:01:45.599 Model run 102 Base
2023-01-13 17:01:45.600   Parameters: 13
2023-01-13 17:01:45.609   Tables: 3
2023-01-13 17:01:45.622   Microdata: 1
2023-01-13 17:01:45.688 Model run 103 Variant
2023-01-13 17:01:45.690   Parameters: 13
2023-01-13 17:01:45.700   Tables: 3
2023-01-13 17:01:45.712   Microdata: 1
2023-01-13 17:01:45.781 Workset 101 Default
2023-01-13 17:01:45.782   Parameters: 13
2023-01-13 17:01:45.798 Done.
```

The console output above was done on Windows. There would be minor cosmetic differences in Linux. Note the use of the global environment variable `OM_ROOT` to ensure that the version of `dbcopy` matches the version of OpenM++ used to build the model.

The `dbcopy` log output shows the extraction of the microdata for the two runs `Base` and `Variant`.

`dbcopy` creates a folder `IDMM` The folder structure of dbcopy output looks like

```
C:\OMPP\MODELS\IDMM\OMPP\BIN\IDMM
├───run.Base
│   ├───microdata
│   ├───output-tables
│   └───parameters
├───run.Variant
│   ├───microdata
│   ├───output-tables
│   └───parameters
└───set.Default
```

Each `microdata` sub-folder contains a file named `Host.csv` containing the microdata of `Host` entities for the run. Had microdata for the `Ticker` actor been requested in the run, a file `Ticker.csv` would also be present. The first few records of `IDMM/run.Base/microdata/Host.csv` look like this:

key	disease_phase
10	DP_SUSCEPTIBLE
11	DP_LATENT
12	DP_IMMUNE

key	disease_phase
13	DP_LATENT
14	DP_IMMUNE
15	DP_LATENT
16	DP_SUSCEPTIBLE
17	DP_LATENT
18	DP_SUSCEPTIBLE
19	DP_IMMUNE
20	DP_IMMUNE

For large output files, one can use the `dbcopy` option `-dbcopy.IdCsv` to output numeric id's instead of alphanumeric codes.

The default microdata key `entity_id` is used in this example. `entity_id` is unique for all entities in a run, and will correspond to the same entity in two `IDMM` runs provided the runs have the same number of entities per replicate and the same number of replicates.

The two files `run.Base/microdata/Host.csv` and `run.Variant/microdata/Host.csv` were imported to Excel, and the 50,000 rows matched one-to-one. Below is an Excel PivotTable (aka cross-tab) which counts the 50,000 `Host` entities at the end of the runs, classified by disease phase in the `Base` run (rows) and disease phase in the `Variant` run (columns).

Base↓/Variant→	DP_IMMUNE	DP_INFECTIOUS	DP_LATENT	DP_SUSCEPTIBLE	All
DP_IMMUNE	24284	806	390	2649	28129
DP_INFECTIOUS	1849	137	92	334	2412
DP_LATENT	2268	94	67	354	2783
DP_SUSCEPTIBLE	13932	421	352	1971	16676
All	42333	1458	901	5308	50000

The lexicographic ordering of disease phase in the table does not follow the ordering in model code, which makes the table harder to interpret. The intuitive order is Susceptible, Latent, Infectious, Immune. That could be addressed by revising the `DISEASE_PHASE` classification codes in `IDMM` model code to align lexicographic order with model code order, e.g.

```
classification DISEASE_PHASE    //EN Disease phase
{
   //EN Susceptible
   DP0_SUSCEPTIBLE,

   //EN Latent
   DP1_LATENT,

   //EN Infectious
   DP2_INFECTIOUS,

   //EN Immune
   DP3_IMMUNE
};
```

Alternatively, the microdata could have been exported using the option `-dbCopy.IdCsv` to output 0,1,2,3 instead of codes in the `csv` files. However, numeric id's in table rows and columns are not informative.

From the table, the level of coherence between `Base` and `Variant` at the end of the simulations is not high. This could be because

- a 10% increase in the duration of immunity is not as minor as one might think *a priori*;
- the increase in duration of immunity is expected to increase the period of epidemic cycles, which would cause epidemic cycles to be out of phase between `Base` and `Variant` at the end of the simulations;
- `IDMM` simulates a highly interacting population which can diverge rapidly from a small initial perturbation;
- simulation divergence is accelerated because `IDMM` does not use entity-specific random number generators for decoherence control.

[back to topic contents]

Worked example 4

This example illustrates run comparison at the microdata level using a large scale complex case-based model (a working version of the Statistics Canada `OncoSimX` model). This example is divided into the following sections:

Example 4 sections

- Summary
- Build steps
- Run steps
- Microdata extraction
- Downstream analysis

[back to topic contents]

Summary

The default microdata key `entity_id` is not suitable for run comparison in `OncoSimX`, so a model-specific definition of `get_microdata_key` was added to model code. A pair of attributes (years lived and health system cost) were output for each `Person` entity at the end of each case. A `Base` run with 500,000 cases and 12 replicates was performed with microdata output enabled, in database mode. A `Variant` run was performed, changing a single scalar parameter. Results for both runs were exported using `dbcopy` to `csv` files and analyzed in Excel to identify all cases which differed between `Base` and `Variant` runs for either of the two attributes.

The mechanical steps in this example are similar to those in the previous example.

[back to example 4 sections]
[back to topic contents]

Build steps

The model code was modified to enable microdata output when the `Person` in each case exits the simulation by adding the following statements to model code.

```
options microdata_output = on;
options microdata_write_on_exit = on;
```

In `OncoSimX` a case contains exactly one `Person` entity, but might contain other entities depending on the simulation, such as one or more `Tumour` entities. Because the built-in attribute `entity_id` is incremented whenever a new entity is created, `entity_id` is unsuitable as a microdata key to match corresponding `Person` entities between two `OncoSimX` runs. However, the built-in attribute `case_id` is suitable as a microdata key for `Person` because it has a one-to-one relationship with the single `Person` entity in each case, and this relationship is robust across runs provided the runs have the same number of cases and replicates. A function definition of `Person::get_microdata_key` was added to model code so that `case_id` is used as the microdata key for `Person` entities instead of `entity_id`:

```
uint64_t Person::get_microdata_key()
{
   return case_id;
}
```

[back to example 4 sections]
[back to topic contents]

Run steps

The model was run using the settings file `OncoSimX/ompp/bin/OncoSimX.ini`, like previous examples.

The following run settings were used for the `Base` run:

```
[OpenM]
SubValues = 12
Threads = 12
RunName = Base

[Parameter]
SimulationSeed = 1
SimulationCases = 500000
MaxConsecutiveHpvTreatmentAllowed = 2

[Microdata]
ToDb = yes
Person = age, cancer_cost_all
```

The parameter `MaxConsecutiveHpvTreatmentAllowed` was chosen arbitrarily for this example. A scalar parameter rather than an array parameter was chosen to make this example simpler, because the value of a scalar parameter can be specified in a model run `ini` file, obviating the need to set up and use a directory for `Variant` parameters which differ from `Base`.

Because the microdata for a `Person` entity is output when a `Person` leaves the simulation at death, the attribute `cancer_cost_all` will contain lifetime cancer-related costs and the `age` attribute will contain the duration of life in years. These two attributes are measures of benefit and cost at the `Person` level. The `case_seed` attribute can be useful to probe a case of interest in a subsequent run, but there is no need to include it in the `Person` microdata attributes because the `key` column already contains the value of `case_seed`, as described above.

For `Variant`, the parameter `MaxConsecutiveHpvTreatmentAllowed` was changed from 2 to 1, and `RunName` was changed to name the run `Variant`:

```
[OpenM]
SubValues = 12
Threads = 12
RunName = Variant

[Parameter]
SimulationSeed = 1
SimulationCases = 500000
MaxConsecutiveHpvTreatmentAllowed = 1

[Microdata]
ToDb = yes
Person = age, cancer_cost_all
```

[back to example 4 sections]
[back to topic contents]

Microdata extraction

After the runs completed, microdata results were extracted from the database using `dbcopy` as in the previous example. Here's the Windows command session:

```
C:\Development\X\models\OncoSimX\ompp\bin>%OM_ROOT%\bin\dbcopy -dbcopy.To csv -dbcopy.ModelName OncoSimX
2023-01-14 18:00:10.259 Model OncoSimX
2023-01-14 18:00:10.392 Model run 102 Base
2023-01-14 18:00:10.392   Parameters: 402
2023-01-14 18:00:16.227     250 of 402: IncidenceRatesHpvMultiplier
2023-01-14 18:00:18.842   Tables: 27
2023-01-14 18:00:25.153     0 of 27: CervicalCancer_TreatmentCost_Table all accumulators
2023-01-14 18:00:31.685     1 of 27: Cervical_Cancer_Cases_PAY_Table all accumulators
2023-01-14 18:00:36.055     7 of 27: Cervical_Cancer_ICER_Table_Discounted all accumulators
2023-01-14 18:00:42.357     26 of 27: Hpv_Screening_Costs_Prov_Table all accumulators
2023-01-14 18:00:43.923   Microdata: 1
2023-01-14 18:00:45.023 Model run 103 Variant
2023-01-14 18:00:45.023   Parameters: 402
2023-01-14 18:00:51.051     250 of 402: IncidenceRatesHpvMultiplier
2023-01-14 18:00:53.794   Tables: 27
2023-01-14 18:01:00.062     0 of 27: CervicalCancer_TreatmentCost_Table all accumulators
2023-01-14 18:01:06.623     1 of 27: Cervical_Cancer_Cases_PAY_Table all accumulators
2023-01-14 18:01:11.144     8 of 27: Cervical_Cancer_LifetimeCost_Table
2023-01-14 18:01:17.385     26 of 27: Hpv_Screening_Costs_Prov_Table all accumulators
2023-01-14 18:01:18.976   Microdata: 1
2023-01-14 18:01:20.115 Workset 101 Default
2023-01-14 18:01:20.116   Parameters: 402
2023-01-14 18:01:26.157     250 of 402: IncidenceRatesHpvMultiplier
2023-01-14 18:01:29.024 Done.
```

The first rows of microdata output for the `Base` run in the file `OncoSimX/ompp/bin/OncoSimX/run.Base/microdata/Person.csv` look like this:

```
key,age,cancer_cost_all
0,79.4991129115706,45100.08867191
1,67.281040126587,2229.937944223
2,87.4865314659319,1670.3732276699
3,0.379665603266858,0
```

The first rows of microdata for the `Variant` run are identical. However, some of the 500,000 microdata output records differ between `Variant` and `Base`.

[back to example 4 sections]
[back to topic contents]

Downstream analysis

An Excel workbook was created and used to

- load the `csv` microdata for `Base` and `Variant` as queries, renaming columns to distinguish `Base` and `Variant`;
- merge the two queries matching on `key` to create a new query with one row for each case and `Base` and `Variant` microdata in distinct columns.
- add a column to the merge query to compute the Variant-Base difference in years lived;
- add a column to the merge query to compute the Variant-Base difference in lifetime cancer-related costs;
- add a column named `Differs` to compute whether a microdata record differed in either years lived or cost between `Base` and `Variant`.

A dynamic filter was applied to the `Differs` column of the Excel table for the merge query to display all records which differed between `Variant` and `Base`. 13 of the 500,000 microdata records differed, as follows:

key	life(base)	cost(base)	life(variant)	cost(variant)	life(delta)	cost(delta)	Differs
26847	82.90	9,099	82.90	10,570	0.0000	1,471	TRUE
59368	89.07	60,812	89.07	61,528	0.0000	717	TRUE
208131	72.68	40,304	98.16	19,647	25.4839	-20,657	TRUE
214559	94.60	31,285	94.60	27,932	0.0000	-3,353	TRUE
229714	86.53	25,446	86.53	13,450	0.0000	-11,996	TRUE
231202	95.18	101,255	95.18	100,388	0.0000	-867	TRUE
247895	97.40	40,914	97.40	9,396	0.0000	-31,518	TRUE
290098	92.17	13,059	92.17	14,461	0.0000	1,402	TRUE
302510	78.51	63,695	78.51	54,770	0.0000	-8,926	TRUE
357201	78.91	8,080	78.91	9,482	0.0000	1,402	TRUE
436603	39.75	112,787	39.75	111,870	0.0000	-916	TRUE
438020	65.36	84,806	63.36	80,545	-2.0000	-4,261	TRUE
447567	94.15	34,830	94.15	32,333	0.0000	-2,498	TRUE

The `key` column contains the value of `case_seed` and could be used to re-simulate any (or all) of these differing cases using Event Trace to explore the different causative pathways taken in the `Base` and `Variant` runs, and how those different pathways affected `Person` attributes.

These differences suggest that it might be interesting to understand how the change in `MaxConsecutiveHpvTreatmentAllowed` from 2 to 1 resulted in

- an additional ~25 years of life for `case_id` 208131,
- both positive and negative changes in health system costs for cases which experienced no change in years lived,
- `case_id` 438020 living an exact integer number of years 2.0000 less in `Variant` compared to `Base`.

Quite possibly all these Base-Variant differences are explained by different but realistic causative pathways taken in the two runs. That could be

verified by comparing the Base and Variant causative pathways for individual differing cases using Event Trace, perhaps by tracing all events, event times, and attribute changes in a differing case and examining differences in the Base and Variant event trace outputs.

This example illustrates how microdata differences between two runs can augment aggregate differences by drilling down to the detail underlying the aggregate differences. It also illustrates how microdata differences from a marginal change to a single model parameter can probe model logic and causative pathways and assist in model validation.

[back to example 4 sections]
[back to topic contents]

Microdata output modes

Two distinct modes are supported: Database mode
and Text mode.

Database mode

- Targeted primarily for use of a production model, to drill down to underlying microdata or to compare two runs at a microdata level.
- uniqueness of key is required
- all microdata output, including from multiple instances and multiple threads, is merged into the model database.
- no run-time event filtering (but can be done in model code with build-time settings).
- `dbcopy` can be used to extract microdata to `csv` files, supports numeric id's or codes.
- `oms` can be used to extract microdata
- will support future functionality for run-time tabulation, including microdata compare (winner-loser).

Text mode

- Targeted primarily to probe a model during development, validation, and debugging
- uniqueness of key is not required
- to trace file or to entity-specific csv files
- runs using multiple instances have distinct csv files for each instance
- multiple threads in an instance share csv files.
- optional event context column
- optional event filtering

Text mode `csv` file names

It is one file per process, all threads do write into the same file. As it is today file name can be:

(a) typical developer / desktop use case: single process, single model run:

ModelName.Entity.microdata.csv

(b) MPI cluster / cloud use case: multiple processes, single model run:

ModelName.Entity.07.microdata.csv

07 is an example of process rank, zero padded It is not limited to 00 - 99, it can be as large as cluster allow us to have, in ComputeCanada can be 5 digits

(c) modelling task run, for example from R or Python using single process:

ModelName.Entity.2022_12_31_22_33_44_981.microdata.csv

2022_12_31_22_33_44_981 is a model run timestamp, time when model run started. Because modelling task run include multiple model runs then each run creates it own microdata cvs file(s)

(d) = c + b: modeling task run in cloud with MPI cluster, it is possible from R on our CPAC cloud:

ModelName.Entity.2022_12_31_22_33_44_981.07.microdata.csv

[back to topic contents]

Microdata output control

This subtopic is divided into the following sections:

Microdata output control sections

- Enabling microdata and controlling warnings
- Weight-enabled models
- Internal attributes
- Attributes with many enumerators

[back to topic contents]

Enabling microdata and controlling warnings

A model is capable of writing microdata if and only if model code contains the following statement:

```
options microdata_output = on;
```

A model with microdata capability will write the following warning to the log whenever it is run:

```
Warning : model can expose microdata at run-time with microdata_output = on
```

If this is not a concern, for example if the model generates entities synthetically, this warning can be disabled by the following statement:

```
options microdata_output_warning = off;
```

[back to microdata output control sections]
[back to topic contents]

Weight-enabled models

A weight-enabled model which is also microdata-enabled will write the following message to the log when run

```
Note : model is weight-enabled and microdata-enabled, include entity_weight in Microdata for downstream weighted operations
```

as a reminder that the attribute `entity_weight` needs to be included in microdata output for downstream weighted tabulation.

[back to microdata output control sections]
[back to topic contents]

Internal attributes

Some internal entity attributes are created by the OpenM++ compiler. For example, the compiler creates an identity attribute to implement the filter of an entity table. These internal entity attributes are normally hidden. They can be made visible, including as microdata, using the following statement:

```
options all_attributes_visible = on;
```

[back to microdata output control sections]
[back to topic contents]

Attributes with many enumerators

Attributes whose type is an enumeration with a large number of enumerators may not be eligible as microdata. For example, the following code fragment declares the `Person` attribute `id_tracker` with type `ID_LIST` which has `5,000,001` possible values (enumerators):

```
range ID_LIST{0, 5000000};

actor Person
{
   ID_LIST id_tracker; //EN Unique identifier of each actor
};
```

If microdata output is enabled, the OpenM++ compiler will emit a warning like

```
PersonCore.mpp(254): warning - attribute 'id_tracker' has 5000001 enumerators making it ineligible as microdata - consider using int.
```

and the attribute `id_tracker` will not be available as microdata at runtime.

However, if `id_tracker` is instead declared to be of type `int` instead of type `ID_LIST`, no warning will be issued and `id_tracker`, with the same integer values assigned in model code, will be available as microdata at runtime.

The maximum number of enumerators for an attribute (of type enumerator) to be eligible as microdata is `1,000`, but can be raised or lowered using the option `microdata_max_enumerators`. For example,

```
options microdata_max_enumerators = 500;
```

will restrict microdata attributes of enumeration type to those with 500 or fewer enumerators. The threshold only applies to attributes declared with enumeration types like `range`. It does not apply to attributes declared with non-enumeration types such as `int`, `counter`, `big_counter`, etc.

Attributes with large numbers of enumerators can cause performance degradation or instability in the microdata viewer and the microdata tabulator due to the large number of cells being manipulated and displayed.

[back to microdata output control sections]
[back to topic contents]

Run-time settings

Run time settings are specified as options, either on the model executable command line or in a model run `ini` file. In an `ini` file, microdata options are in the `[Microdata]` section. On the command line, they are given like `-Microdata.Person age`.

The following table lists all microdata run-time settings with an example and a short description.

Option	Example	Description
entity	`Person` `=` `ageGroup,sex,time`	Store the named attributes for the specified entity kind, e.g. the attributes `ageGroup`, `sex`, and `time`, for `Person` entities.
entity	`Person = All`	Store all non-internal attributes of `Person` entities.
`ToDb`	`true`	Write microdata entity attributes into database. Important: each microdata entity *must* have a unique key. Default is `false`.
`ToCsv`	`true`	Write microdata entity attributes and events (if enabled) into `csv` file(s). each microdata entity is written in its own file. Default is `false`.
`UseInternal`	`true`	Store all non-internal attributes of all entities. NOT recommended for production, use for debug only. Default is `false`.
`CsvDir`	`path/to/some/directory`	Directory where microdata `csv` file(s) are written, must be an existing directory. Default is the current directory.
`ToTrace`	`true`	Write microdata entity(s) attributes and events (if enabled) to model Trace output. Trace must be enabled to produce any output. Default is `false`.
`Events`	`Birth,Union,Death`	Write selected events into Trace or `csv` file.
`CsvEventColumn`	`true`	If `true` then write event name into `csv` file. Default is `false`.

For a complete example of a run `ini` file, including the `[Microdata]` section, see OpenM++ ini-file run options.

[back to topic contents]

Build-time settings

Build-time settings which enable a model to output microdata are described in Enabling microdata output. Other build-time options can output microdata during the simulation of each entity. The available options are:

Option	Default	Description
`microdata_write_on_enter`	`off`	microdata is written when an entity enters the simulation, before any event occurs in the entity.
`microdata_write_on_exit`	`off`	microdata is written when an entity exits the simulation.
`microdata_write_on_event`	`off`	microdata is written after an event occurs in an entity.

These options can be combined. If none of these options are `on` no microdata will be written unless model code does so explicitly by calling or hooking the built-in function `write_microdata`.

Note that attributes of an entity can change due to events in other linked entities in a model with interacting entities. So, even if `microdata_write_on_event` is `on`, changes in attributes of an entity can be absent from microdata output for that entity. For example, in `IDMM`, if an infectious `Host` A infects `Host` B through A's social contacts, the event associated with the infection occurs in A and not in B. If one wanted to output `Host` microdata at the moment of infection, one could do so by calling `output_microdata` explicitly in model code.

[back to topic contents]

Writing microdata from model code

Under construction

Microdata can be written by calling the built-in entity function `write_microdata()` from model code, either directly or by using a `hook` statement.

If a model is not enabled for microdata, calls to `write_microdata` have no effect.

Modgen-specific: The Modgen build of a cross-compatible model inserts a do-nothing version of `write_microdata()` into the Modgen-generated C++ code. This allows use of `write_microdata` in model code without producing C++ build errors in the Modgen build of a x-compatible model.

[back to topic contents]

The microdata key

A key is a unique identifier used to match entities or microdata records across runs. It is a 64-bit value of C++ type `uint64_t`.

The *key for an entity* is returned by the entity member function `get_entity_key()`. If this function is not defined in model code, the OpenM++ compiler will provide a definition which returns the value of the built-in attribute `entity_id`. The entity key is described further here.

The *key for a microdata output record* is produced by the entity member function `get_microdata_key()`. If this function is not defined in model code, the OpenM++ compiler will provide a definition which computes and returns the following value:

```
10000 * get_entity_key() + om_microdata_counter
```

where `om_microdata_counter` is an internally-maintained counter of output microdata records for each individual entity. This formula produces unique microdata keys because it combines the unique `entity_id` with an entity-specific counter of microdata records output for that entity. Uniqueness is guaranteed provided that fewer than 10,000 microdata records are output for a single entity.

For example, if microdata is output at each event using the `microdata_write_on_event` option, the default microdata key would be

`entity_id`	Event	`get_microdata_key()`
42	first	420001
42	second	420002
101	first	1010001
101	second	1010001

The OpenM++ compiler generates C++ code to create and maintain `om_microdata_counter` only if the `get_microdata_key()` function has not been defined for that kind of entity in model code.

Note that the entity key from `get_entity_key()` can be used both to calculate the result of `get_microdata_key()` and to support local random streams in model architectures which use both.

Uniqueness of the microdata key is enforced for Database mode, but is not enforced for Text mode.

A model run-time error will occur if uniqueness of the microdata key is violated in Database mode, with log output similar to the following:

```
2023-01-24 12:22:27.202 Writing microdata into database, run: 102
2023-01-24 12:22:28.525 : UNIQUE constraint failed: Host_g732a1637.run_id, Host_g732a1637.entity_key
2023-01-24 12:22:28.528 Error at microdata: 2100, 100, 3
2023-01-24 12:22:28.559 DB error: UNIQUE constraint failed: Host_g732a1637.run_id, Host_g732a1637.entity_key
```

The line `Error at microdata: 2100, 100, 3` indicates that the record `{2100, 100,3}` violated key uniqueness. The first value is the non-unique `key` value, which is `2100` in this example. The following values are the other attributes of the microdata record with the non-unique `key`

The case-based model in Example 4 supplies a custom implementation of `get_microdata_key()` to correctly match `Person` microdata results between two runs, because in that model the number of secondary entities in a case can vary between two runs. The time-based model in Example 3 uses the standard implementation of `get_microdata_key()` because there are no additions to the starting population of `Host` entities created at the beginning of a run, and runs of equal size are being compared.

The following hypothetical definition of `get_microdata_key()` uses the helper function `xz_crc64` to combine the value of `get_entity_key()` and `report_time` to create the microdata key. `xz_crc64` creates a 64-bit key using the `crc-64` open source checksum (hash) algorithm, and can use one value or combine multiple values together using successive calls.

```
uint64_t Host::get_microdata_key()
{
   uint64_t key64 = 0;
   auto entity_key = get_entity_key();
   key64 = xz_crc64((uint8_t*)&entity_key, sizeof(entity_key), key64);
   key64 = xz_crc64((uint8_t*)&report_time, sizeof(report_time), key64);
   return key64;
}
```

This definition might be used in a model which outputs a microdata record for each entity at each `report_time`, if the number of entities might vary from one run to another due to parameter differences (e.g. fertility).

For a model based on a microdata input file where each input record has a unique personal identification number `person_id`, and in which only a single microdata record is output for each `Person`, a suitable definition might look like:

```
uint64_t Person::get_microdata_key()
{
   uint64_t key64;
   key64 = person_id;
   return key64;
}
```

[back to topic contents]

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
- Doc folder and documentation 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 files are 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.
- `ompp_options.ompp` Model-specific source file containing `options` statements which specify commonly modified model options, such as Model Documentation options.

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]

Doc folder and documentation files

The human language documentation of an OpenM++ model can be embedded in the model source code in Label and Note comments or in files in a single model documentation folder, e.g. `RiskPaths/doc` for the `RiskPaths` model. Each model documentation file has a name and extension which determine its language and role when model documentation is built, for example:

- `NOTE.SymbolName.EN.md` Markdown file documenting the model symbol `SymbolName` in English.
- `NOTE.SymbolName.FR.md` Markdown file documenting the model symbol `SymbolName` in French.
- `LABEL.SymbolName.EN.txt` Text file containing the label of the model symbol `SymbolName` in English.
- `LABEL.SymbolName.FR.txt` Text file containing the label of the model symbol `SymbolName` in French.
- `Home.EN.md` Markdown file containing the Home topic of the Authored Model Documentation in English.
- `Home.FR.md` Markdown file containing the Home topic of the Authored Model Documentation in French.
- `TopicName.EN.md` Markdown file containing the autonomous authored topic `TopicName` in English.
- `TopicName.FR.md` Markdown file containing the autonomous authored topic `TopicName` in English.
- `*.pdf` Auxiliary downloadable PDF documentation file

See Model Documentation for more about model documentation.

[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 optional include file `omc/optional_IDE_helper.h` at the top of the model code file, as follows:

```
#include "omc/optional_IDE_helper.h" // help an IDE editor recognize model symbols
```

Modgen-specific: The optional helper include file `omc/optional_IDE_helper.h` is x-compatible and will not interfere with a Modgen build.

[back to topic contents]

Model Documentation

This is the main topic on model documentation.

Related topics

- Authored Model Documentation: The autonomous authored component of model documentation
- Generated Model Documentation: The Symbol Reference component of model documentation
- Model Languages: The human languages supported by a model
- Model Symbols: Symbols in model code and in the user interface
- Symbol Labels and Notes: Human-language labels and notes for model symbols

Topic contents

- Introduction and outline
- Use cases Use cases for model documentation: end users, model developers, content developers
- The two components of model documentation Autonomous authored and the generated Symbol Reference
- Building model documentation How to build (or not) model documentation and control its content
- Using model documentation UI access, stand-alone file, searching, printing
- Exploring internals using doxygen Going down the rabbit hole of OpenM++ internals

Introduction and outline

For an immediate hands-on tour of model documentation, build the `RiskPaths` model, open the model UI, and click on the 'book' icon. For a model or content developer perspective, open the `RiskPaths` project and explore the contents of the `doc` sub-directory containing markdown content. The example `RiskPaths` model documentation is configured for model and content developers.

Model documentation has multiple audiences and multiple use-cases, and a one-size-fits-all approach is unlikely to match all of these needs well. The architecture of Model Documentation in OpenM++ is designed for flexibility and easy reconfiguration to meet one need or another, for the same model.

This topic starts with a general exposition on the principal use cases for model documentation. It then describes the two main components of model documentation and how they are structured into topics. The next two subtopics describe how to build and use model documentation. A final stand-alone subtopic describes how `doxygen` can be used to explore model internals, including components normally hidden from model developers.

[back to topic contents]

Use cases

Model documentation has several different audiences:

- model users
- model developers
- content developers

Model users

Model users need to understand the model in general, including its

- substance
- methods
- intended uses

- limitations

Users who run the model also need to understand the meaning and proper use of the model symbols exposed through the user interface, run downloads, or through external tools like R or Python. Those symbols consist of exposed

- input parameters
- output tables
- entity attributes in microdata
- enumerations associated with exposed parameters, tables, and attributes.

Typically, only a subset of model symbols are exposed to model users. Model users access model documentation through the model UI, or perhaps through a stand-alone PDF or HTML 'User Edition' file built and provided by a model developer.

[back to use cases]
[back to topic contents]

Model developers

Model developers need to understand all aspects of the model:

- probe the model to understand or debug issues
- understand internal relationships and dependencies among symbols and model code
- develop new functionality coherent with existing code

Model developers access model documentation through the model UI, or perhaps by directly opening a previously built HTML 'Developer Edition' file. They may also access model documentation through an IDE when working with the model source code. Typically, model developers access all of the models symbols, including symbols not directly relevant to model users such as entity sets, functions, and model code modules, as well as parameters, tables, entity attributes, and enumerations not visible to users in the UI.

[back to use cases]
[back to topic contents]

Content developers

Content developers use model documentation to review and author model documentation, focusing on

- introducing the model to new users
- presenting the subject matter of the model
- reviewing the coherence and coverage of model documentation

[back to use cases]
[back to topic contents]

The two components of model documentation

Model documentation consists of two components which can work independently or synergistically:

1. A configurable Symbol Reference of generated cross-referenced content on parameters, tables, enumerations, etc., available in either a User Edition (default) or a Developer Edition.
2. Optional autonomous authored model documentation, organized as a collection of distinct topics, which can link to each other, to topics in the Symbol Reference, or to optional downloadable supporting files.

Model documentation can consist of one of these two components, or both.

Each component consists of a collection of topics, where each topic can be the target of a link from elsewhere in the model documentation.

A topic in the Symbol Reference is either a model symbol (the topic name / link target is the symbol name), or one of several fixed topics used to navigate the Symbol Reference. The main topic of the Symbol Reference contains some tombstone information about the model and a table of links to several navigation aid topics. By default, the Symbol Reference is targeted to a model user, so it documents only those symbols which are exposed in the model UI.

A topic in autonomous authored documentation is a `.md` markdown file in the model's `doc` sub-directory (the topic name / link target is the stem of the filename of the corresponding `.md` markdown file). If present, the `Home` authored document becomes the main (first) topic of model documentation. Other autonomous authored topics follow in lexicographic order by topic name (same order as filename).

[back to topic contents]

Building model documentation

By default, model documentation is built and assembled as an intrinsic step in the model build. Building model documentation for a large model can take a noticeable amount of time, so a model developer might want to turn it off to speed development cycles, and turn it back on from time to time to refresh the Developer Edition of the Symbol Reference for their own use, or to build a final User Edition version of model documentation for a model release.

In Visual Studio, the Model project OpenM++ property page contains an option to turn building model documentation on or off:

In development environments using `make`, setting the `MODEL_DOC_DISABLE` variable will disable the model documentation build step.

The most recently built version of model documentation will continue to be available from the model UI, even after the model is rebuilt. If model documentation is disabled in the build, its content may not reflect the model code, and it may differ from context-sensitive help displayed in the UI.

Symbol labels and notes for symbols exposed to the user through the UI are always published, independent of build settings for model documentation. The model UI uses those labels and notes to display up-to-date context-specific information on exposed parameters, tables, attributes, and for any enumerations used by those exposed symbols.

Which of the two documentation components are built and assembled into the final model documentation can be controlled using the following options in model code, e.g. in the model code module `code/ompp_options.ompp`:

```
options authored_documentation = on;
options generated_documentation = on;
```

By default, both options are `on`, and if the model has no autonomous authored content, model documentation will consist of only the generated Symbol Reference component.

The content of the generated Symbol Reference, including the choice of User Edition or Developer Edition, can be controlled through options in model code, as described in customizing the Symbol Reference.

[back to topic contents]

Using model documentation

For each model language, the two components of model documentation are built and assembled into one `.html` file for each model language. The UI links to this file when a user requests model documentation by clicking the 'book' icon. The 'book' icon is available in several places in the UI:

- the landing page which lists available models
- the title bar displayed at the top of all pages
- the information pop-up for a symbol

Model documentation can take a noticeable time to load in a browser if the model is large and complex, and if the Developer Edition of the Symbol Reference is selected.

From a browser, the entire model documentation can be searched using the browser's `Find on page` command or `Ctrl-F` key combination to identify all occurrences of a word or phrase in the model documentation.

From a browser, the model documentation can be converted from `.html` to `.pdf` format by invoking the browser's `Print` command or `Ctrl-P` key combination and selecting `pdf` as the target printer. Model documentation is designed so that each topic starts on a new page in the resulting `.pdf` file (only tested on the Microsoft Edge browser).

The `.html` files containing the model documentation can be accessed directly in the file system used to build the model. They are located in a `doc` sub-directory of the `bin` directory of the model (not to be confused with the `doc` sub-directory of the model root, which contains authored content). For `RiskPaths`, the files are

`RiskPaths/.../bin/doc/RiskPaths.doc.EN.html` (English)
and
`RiskPaths/.../bin/doc/RiskPaths.doc.FR.html` (French)

Label and note information for symbols exposed in the UI are always written to the model database for access by the UI, external utilities like `dbcopy`, the `oms` business layer, and (possibly) external packages written `R` or `python` using `oms`.

[back to topic contents]

Exploring internals using doxygen

The wiki subtopic Background on doxygen mentions that the doxygen application can generate hyperlinked HTML documentation for a C++ project. After the OpenM++ compiler has run and produced the C++ code to implement a model, a complete C++ project for that model exists. Doxygen can be run with this project as input to create hyperlinked HTML documentation of the C++ code of the model. An example doxygen input file for `RiskPaths` is located at `OM_ROOT/models/RiskPaths.doxyfile`. Below is a screenshot of a browser displaying the C++ documentation produced by doxygen for `RiskPaths`:

The screenshot shows information for the `Person` member function `timeUnion1DissolutionEvent()` which is an event time function supplied in model code. Note that hovering over the symbol `union_duration` in the model source code gives summary information about the symbol, a bit like the display in the Visual Studio IDE. Note also the References section below the function body, which lists all non-local variables used in the function with each hyperlinked to its section elsewhere in the doxygen-generated HTML documentation.

The documentation generated by doxygen for RiskPaths is based not just on model code. It is also based on C++ code generated by the OpenM++ compiler and fixed framework OpenM++ code. That code is rarely pertinent to a model developer. Future enhancements may remove those superfluous portions from doxygen output and add customized doxygen sections targetted to model developers.

[back to topic contents]

Model Languages

The human languages supported by a model.

This topic is under construction and/or revision.

Related topics

- Authored Model Documentation: The autonomous authored component of model documentation
- Generated Model Documentation: The symbol reference component of model documentation
- Model Symbols: Symbols in model code and in the user interface
- Symbol Labels and Notes: Human-language labels and notes for model symbols

Topic contents

- Introduction and outline
- The `languages` statement

Introduction and outline

Content to follow.

[back to topic contents]

The `languages` statement

Each human language supported by a model has an associated *language code* given in the `languages` statement of the model.

A model is required to have a `languages` statement which declares at least one language.

For example,

```
languages
{
  EN, // English
  FR  // Français
};
```

declares two languages `EN` and `FR`. These codes can be used in model code to provide language-specific documentation of model symbols. OpenM++ imposes no limitations on language codes, but it is strongly recommended to use the two-letter prefix of standard locale codes, as in the example above. Using upper case language codes helps them stand out in model code. So, the suggested code for Spanish is `ES`, and for German `DE`. The OpenM++ UI renders model-independent text, e.g. the word `Run`, using a human-language specific library. Libraries for `EN` and `FR` are currently available. If there is no library for a language specified in a model, the UI will display model-specific content in that language, but will fall back to one of the available libraries for model-independent text.

The first language listed in the `languages` statement is the *default language* of the model, which in this example is `EN`. The default language of the model is usually the language in which the source code is written, i.e. the choice of symbol names and the language of explanatory code comments. The language presented initially in the UI might not be the default language, but instead the language of the locale on the end user's device. An end user can switch languages dynamically in the UI.

The trailing comment after each language code provides a language-specific label for the language code declared previously in the line. The UI uses these labels to display human-language versions of the model's available languages when a user selects or changes the UI language.

[back to topic contents]

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]

Model Metrics Report

This topic describes and illustrates the Model Metrics Report produced by the OpenM++ compiler. The report summarizes code inputs and code outputs, model inputs and model outputs, model symbols, and maintained dependencies.

Related topics

- Model Code: How model code is organized into modules, syntactic islands, and C++
- Model Resource Use: Activating and interpreting information about model resource use

Topic contents

- Introduction
- Code input
- Code output
- Model input
- Model output
- Model Symbols
- Published Symbols
- Maintained dependencies

Introduction

The OpenM++ compiler (`omc`) produces the Model Metrics Report in the file `MODEL/.../src/ModelMetrics.txt`. The report consists of 7 summary tables about the model's

- source code base
- user interface
- symbols
- implicit causal web

The Model Metrics Report is about the model, not about a run of the model. To produce metrics for a run of the model, see Model Resource Use.

The report begins with a title line like

```
Model Metrics for RiskPaths 2023-10-13
```

which gives the model name and the date of compilation.

The remainder of this topic describes each of the 7 tables in turn using the `RiskPaths` model as an example.

[back to topic contents]

Code input

The `CODE INPUT` table is a summary of the model code processed by `omc`.
Here it is for `RiskPaths`:

```
+-----------------------------------+
| CODE INPUT (model source code)    |
+-----------------+-------+---------+
| Source          | Files |   Lines |
+-----------------+-------+---------+
| Model-specific  |    14 |    1504 |
|   Islands       |       |     286 |
|   C++           |       |    1218 |
| Common          |     9 |    1707 |
|   Islands       |       |      11 |
|   C++           |       |    1696 |
| Both            |    23 |    3211 |
|   Islands       |       |     297 |
|   C++           |       |    2914 |
+-----------------+-------+---------+
| Total           |    23 |    3211 |
+-----------------+-------+---------+
Note: Does not include parameter data.
```

The table groups the model source code files into two categories:

- model-specific files in the `MODEL/code` folder
- common files supplied by OpenM++ and incorporated into the model by `use` statements

The model-specific `RiskPaths` source code consists of 1,504 lines in 14 `.mpp` and `.ompp` files. An example of one of those 14 files is `RiskPaths/code/PersonCore.mpp`, which implements core functionality for the `Person` entity.

The common source code in `RiskPaths` consists of 1,707 lines in 9 `.ompp` files specified by `use` statements. An example of one of those 9 files is `OM_ROOT/use/random/random_lcg41.ompp`, which implements a family of simple and fast random number generators.

Common source code files are specified in `use` statements, which in `RiskPaths` are in the model-specific source code file `RiskPaths/code/ompp_framework.ompp`. Common source code files can also contain `use` statements.

The `MODEL CODE` table assigns each line of a source code file to one of 2 categories:

- Syntactic island
- C++ code

A syntactic island is a section of declarative code written in the ompp language. Here's an example from `RiskPaths` - the declaration of the table `T02_TotalPopulationByYear`:

```
table Person T02_TotalPopulationByYear  //EN Life table
{
  //EN Age
  integer_age *
  {
    unit,      //EN Population start of year
    duration()  //EN Average population in year
  }
};
```

A syntactic island starts with a keyword, e.g. `table`, and ends with a matching `;`.

Source code which is not part of a syntactic island is C++ code. Model C++ code implements C++ functions associated with events, derived parameters, and derived tables. It can also implement other functions to support the model.

Here's an example of model C++ code from `RiskPaths` - the C++ function which implements the `DeathEvent`:

```
void Person::DeathEvent()
{
  life_status = LS_NOT_ALIVE;
  Finish();
}
```

For more on model source code, see Model Code.

The `CODE INPUT` table does not report on parameter value files read by `omc` to publish the starting Default scenario/set for the model. For `RiskPaths` those are the 2 files `RiskPaths.dat` and `Framework.odat` in the folder `RiskPaths/parameters/Default`.

[back to topic contents]

Code output

The `CODE OUTPUT` table contains summary information about C++ code generated by `omc`.
Here it is for `RiskPaths`:

```
+----------------------------------------+
| CODE OUTPUT (generated C++)            |
+-----------------------+-------+---------+
| Description           | Files |   Lines |
+-----------------------+-------+---------+
| Header (.h)           |     3 |    1395 |
| Implementation (.cpp) |     1 |    3766 |
+-----------------------+-------+---------+
| Total                 |     4 |    5161 |
+-----------------------+-------+---------+
Note: Table does not include C++ initializers for burned-in parameter data.
```

`omc` uses the *declarative* model specification in syntactic islands to generate *procedural* C++ code which implements the model specification. For `RiskPaths`, the 297 lines in syntactic islands which specify the model are used by `omc` to generate the 5,161 of C++ code which implement the model. The large difference between the # of lines of input declarative code and the # of lines of output procedural code reflects the conciseness and power of the ompp declarative language.

Generally, model developers don't need to examine generated C++ code because it just works, but here's a peek at it anyway. The following 2 extracts show the code generated by `omc` for the internal function `om_side_effects_age_status`, one of the C++ functions used to implement `RiskPaths`.

The function is declared in the generated header file `RiskPaths/.../src/om_declarations.h` and looks like this:

```
// model entity classes
class Person : public Entity<Person>
{
public:
    ...
    /// Implement side effects of changing age_status in entity Person.
    void om_side_effects_age_status(int om_old, int om_new);
    ...
};
```

The function is defined in the generated implementation file `RiskPaths/.../src/om_definitions.cpp` and looks like this:

```
void Person::om_side_effects_age_status(int om_old, int om_new)
{

    // Code Injection: group=8, injector=formation_hazard
    // Maintain identity for 'formation_hazard'
    formation_hazard_update_identity();

    // Code Injection: group=8, injector=preg_hazard
    // Maintain identity for 'preg_hazard'
    preg_hazard_update_identity();

    // Code Injection: group=10, injector=FirstPregEvent
    // Recalculate time to event FirstPregEvent
    if (om_active) om_FirstPregEvent_om_event.make_dirty();

    // Code Injection: group=10, injector=Union1FormationEvent
    // Recalculate time to event Union1FormationEvent
    if (om_active) om_Union1FormationEvent_om_event.make_dirty();
}
```

The generated function `om_side_effects_age_status` handles the side-effects of a change in the `age_status` attribute of the `Person` entity. It is automatically called if the value of `age_status` changes during an event in a run, and when called it

- updates the identity attributes `formation_hazard` and `preg_hazard` because their declarations in syntactic islands use `age_status`.
- marks for recalculation the future times of the events `FirstPregEvent` and `UnionFormationEvent` because the time functions of those events use `age_status`, which could affect those future times.

These 4 actions in `om_side_effects_age_status` are examples of Maintained dependencies, which have a dedicated table of their own in the Model Metrics report.

This example shows that the C++ code generated by `omc` contains comments as well as code and is formatted for readability. This can help to understand the generated code if there's ever a need to examine or trace into it in a debug session of a model.

`omc` also generates, in the output C++ header file `MODEL/.../src/om_declarations.h`, doxygen brief descriptions for model symbols such as classifications, ranges, partitions, parameters, and attributes. For example, the generated doxygen brief description for the `age_status` attribute looks like this:

```
/// attribute(identity) int: Current age interval
age_status_om_type age_status;
```

The generated doxygen brief description which starts with `///` says that `age_status` is an attribute, more specifically an identity attribute, that its type is `int`, and its label in the model's default human language is `Current age interval`. Doxygen brief descriptions are automatically recognized by the Visual Studio and VSCode IDEs, and those IDEs display a popup with the doxygen brief description for a symbol when the cursor is hovered over that symbol name in model C++ source code. For more, see Doxygen brief descriptions for model symbols.

`omc` transforms burned-in parameter values to C++ initializers in the output file `MODEL/.../src/om_fixed_parms.cpp`. The `CODE OUTPUT` table does not report on this file or its contents.

[back to topic contents]

Model input

The `MODEL INPUT` table provides summary information about the model's input parameters. It reports counts of parameters and counts of parameter cells.

`MODEL_INPUT` classifies parameters into 4 mutually exclusive categories:

- Visible parameters, which are immediately available in the UI.
- Hidden parameters, which are made visible in the UI when the user clicks a reveal button.
- Burned-in parameters, which have fixed values and are absent from the model database and UI. A parameter is burned into the model executable if its Default value file is located in the `Fixed` folder, or if it was removed from the model database and UI by a `suppress_parameters` or `retain_parameters` statement.
- Derived parameters, which are computed by model-specific code during run initialization.

Here's the `MODEL INPUT` table for `RiskPaths`:

```
+----------------------------------+
| MODEL INPUT (parameters)         |
+---------------+-------+----------+
| Kind          | Count |    Cells |
+---------------+-------+----------+
| Visible       |     2 |       18 |
| Hidden        |     7 |      133 |
| Burned-in     |     0 |        0 |
| Derived       |     0 |        0 |
+---------------+-------+----------+
| Total         |     9 |      151 |
+---------------+-------+----------+
Note: Burned-in includes fixed and suppressed parameters.
```

The table shows that `RiskPaths` presents a highly simplified UI to users by hiding all but 2 parameters. The 2 visible parameters contain 18 values (they're arrays). The remaining parameters are hidden but can be examined and modified by a user by clicking the reveal button in the UI. This design reflects that `RiskPaths` was used as a hands-on simple example in a demography course.

[back to topic contents]

Model output

The `MODEL OUTPUT` table provides summary information about the model's output tables. It reports counts of tables and counts of table cells.

`MODEL_OUTPUT` classifies tables in two ways:

Entity/Derived:

- Entity tables are declared in model-specific syntactic islands. They are computed dynamically during a run.
- Derived tables are computed by model-specific C++ code after the simulation phase completes.

Visible/Hidden:

- Visible tables are immediately available in the UI after a run completes.
- Hidden tables are made visible in the UI when the user clicks a reveal button.

Here's the `MODEL OUTPUT` table for `RiskPaths`:

```
+-----------------------------------+
| MODEL OUTPUT (tables)             |
+---------------+-------+----------+
| Kind          | Count |    Cells |
+---------------+-------+----------+
| Entity        |       |          |
|   Visible     |     2 |      205 |
|   Hidden      |     5 |      148 |
| Derived       |       |          |
|   Visible     |     0 |        0 |
|   Hidden      |     0 |        0 |
| Both          |       |          |
|   Visible     |     2 |      205 |
|   Hidden      |     5 |      148 |
+---------------+-------+----------+
| Total         |     7 |      353 |
+---------------+-------+----------+
Note: Cells includes margins and expression dimension.
```

For didactic purposes, `RiskPaths` shows only 2 key tables in the UI when a run completes. However, all tables in `RiskPaths` can be made visible and explored in the UI. No derived tables were required in `RiskPaths`, so those rows are zero in the report.

[back to topic contents]

Model Symbols

The `MODEL SYMBOLS` table reports counts of all symbols in syntactic islands by category, together with counts of associated labels and notes in the model's default human language.

Some model symbols are given names in the model code, such as the table `T02_TotalPopulationByYear` in the example in Code input. Some model symbols are declared positionally in the model code with no name, such as the expression dimension of that table. Other model symbols are declared when used, such as the derived attribute `duration(parity_status,PS_CHILDLESS)` which automatically maintains a running sum of the time a `Person` has spent in the `PS_CHILDLESS` state.

Here's the `MODEL SYMBOLS` table for `RiskPaths`:

```
+-----------------------------------------------+
| MODEL SYMBOLS                                 |
+-------------------------+-------+-------+-------+
| Description             | Count | Label |  Note |
+-------------------------+-------+-------+-------+
| Language (human)        |     2 |     0 |     0 |
| Enumeration             |       |       |       |
|   Classification        |     4 |     4 |     0 |
|     Level               |    12 |    12 |     0 |
|   Range                 |     1 |     1 |     0 |
|   Partition             |     4 |     4 |     0 |
|   Aggregation           |     0 |     0 |     0 |
| Input/Output            |       |       |       |
|   Parameter             |     9 |     9 |     0 |
|     Dimension           |     7 |     7 |     0 |
|     Group               |     3 |     3 |     0 |
|   Table                 |     7 |     7 |     0 |
|     Dimension           |    13 |     6 |     0 |
|     Expression          |    13 |    13 |     0 |
|     Group               |     3 |     3 |     0 |
|   Import                |     0 |     0 |     0 |
| Entity                  |       |       |       |
|   Kind                  |     1 |     0 |     0 |
|   Event                 |     7 |     0 |     0 |
|   Attribute             |    38 |    13 |     0 |
|     Built-in            |     6 |     0 |     0 |
|     Simple              |     5 |     5 |     0 |
|     Identity            |    13 |     8 |     0 |
|     Derived             |    14 |       |       |
|     Link                |     0 |     0 |     0 |
|     Multilink aggregate |     0 |       |       |
|   Function              |     2 |     2 |     2 |
|   Multilink             |     0 |     0 |     0 |
|   Array                 |     0 |     0 |     0 |
|   Foreign               |     0 |     0 |     0 |
| Entity set              |     0 |     0 |     0 |
|   Dimension             |     0 |     0 |     0 |
+-------------------------+-------+-------+-------+
| Total                   |   126 |    84 |     2 |
+-------------------------+-------+-------+-------+
| Supplementary Info      |       |       |       |
|   Random streams        |     6 |       |       |
|   Eligible microdata    |     0 |       |       |
+-------------------------+-------+-------+-------+
Notes: Parameter includes derived parameters.
       Table > Dimension includes the expression dimension.
       Entity > Attribute > Identity includes those generated from filters.
       Entity > Function does not include event time and implement functions.
```

The `RiskPaths` model is relatively simple, and contains only 126 symbols in syntactic islands.

The `Eligible microdata` row has a count of 0 because `RiskPaths` does not enable microdata output. For more, see Microdata Output.

[back to topic contents]

Published Symbols

The `PUBLISHED SYMBOLS` table is similar to `MODEL SYMBOLS` but reports only on symbols which are published to the model database and visible to model users in the UI, exported model metadata, or run data. Model code can suppress or retain parameters and tables to create a trimmed down model for publishing, while full preserving model logic. For more on that, see the subtopic Model trim down.

Here's the `PUBLISHED SYMBOLS` table for `RiskPaths`:

```
+-------------------------------------------+
| PUBLISHED SYMBOLS                         |
+-------------------+-------+-------+-------+
| Description       | Count | Label |  Note |
+-------------------+-------+-------+-------+
| Language (human)  |     2 |     0 |     0 |
| Enumeration       |       |       |       |
|   Classification  |     2 |     2 |     0 |
|     Level         |     8 |     8 |     0 |
|   Range           |     1 |     1 |     0 |
|   Partition       |     4 |     4 |     0 |
| Input/Output      |       |       |       |
|   Parameter       |     9 |     9 |     0 |
|     Dimension     |     7 |     7 |     0 |
|     Group         |     3 |     3 |     0 |
|   Table           |     7 |     7 |     0 |
|     Dimension     |    13 |     6 |     0 |
|     Expression    |    13 |    13 |     0 |
|     Group         |     3 |     3 |     0 |
|   Import          |     0 |     0 |     0 |
+-------------------+-------+-------+-------+
| Total             |    72 |    63 |     0 |
+-------------------+-------+-------+-------+
Note: Table > Dimension includes the expression dimension.
```

The table shows that `RiskPaths` supplies labels for almost all published model symbols, which can help model users navigate and understand model inputs and outputs. `RiskPaths` eschews the use of notes, which would have been viewable with a click in the UI, providing additional information about the model's parameters and tables.

This table reveals a couple of minor issues in `RiskPaths`. First, human-language labels for the 2 languages declared in `RiskPaths` (`EN` and `FR`) are absent for the model's default human language. This causes OpenM++ to present language codes rather than language names in the UI for switching languages. Second, 7 table dimensions lack labels in the model's default language, which causes `omc` to generate fall-back labels mechanically.

[back to topic contents]

Maintained dependencies

As described in Code input, the ompp language is *declarative*. Here again is the declaration of the table `T02_TotalPopulationByYear`:

```
table Person T02_TotalPopulationByYear  //EN Life table
{
  //EN Age
  integer_age *
  {
    unit,      //EN Population start of year
    duration()  //EN Average population in year
  }
};
```

The resulting table has 101 rows for integer age and 2 columns for the computed expressions. Whenever a `Person` has a birthday and `integer_age` changes, the cell in `T02_TotalPopulationByYear` to which they are contributing changes and the table must be updated. In other words, there is a dependency between `integer_age` and `T02_TotalPopulationByYear`.

The example in Code output illustrates other kinds of dependency, where changes in the `age_status` attribute requires updating the identity attributes `formation_hazard` and `preg_hazard` and and recalculating the events `FirstPregEvent` and `Union1FormationEvent`.

To implement the model specification, `omc` generates C++ code which maintains all such dependencies, both direct and indirect. The `MAINTAINED DEPENDENCIES` table groups and counts these dependencies by category.
Here's it is for `RiskPaths`:

```
+----------------------------+-------+
| MAINTAINED DEPENDENCIES            |
| (in generated C++ runtime code)    |
+----------------------------+-------+
| Dependency                 | Count |
+----------------------------+-------+
| Reciprocal link            |     0 |
| Attribute maintenance      |       |
|   Identity                 |    18 |
|   Derived                  |    23 |
|   Multilink aggregate      |     0 |
| Table dimension/filter     |     8 |
| Set dimension/filter/order |     0 |
| Event maintenance          |    17 |
+----------------------------+-------+
| Total                      |    66 |
+----------------------------+-------+
```

The table shows that the implicit causative web in `RiskPaths` contains 17 dependencies between attributes and events.

`RiskPaths` is a simple model with only 286 lines of model code in syntactic islands. Nevertheless, it has 66 dependencies which are automatically maintained by the C++ code generated by `omc`.

[back to topic contents]

Model Resource Use

This topic describes how to activate and interpret reports on run-time computation and memory use.

Related topics

- Model Code

Topic contents

- Introduction and Background
- Syntax and Use How to activate and use
- Illustrated Reference Illustrated reference of each report use table
- Appendix The complete report used in the illustrations (over 500 lines)

Introduction and Background

OpenM++ models can be demanding applications for processing and memory use. The `resource_use` option collects and reports information which can help model developers identify areas where efficiency can be improved. The reports can also sometimes identify model anomalies revelaed by unusually high (or low) resource use. For example, the reports can identify an event never occurs, or an an event with an anomalously high event frequency per entity.

The following subsections provide background on entities, multilinks, entity tables, sets, and events which may be helpful in understanding resource use reports.

Entities

An entity of a given type such as a `Person` in the RiskPaths model consists of a block of memory which holds the values of attributes and other data members. Each type of entity has a fixed size. For efficiency, entities which leave the simulation are placed in a pool for potential reuse. As a result, the number of activated entities (entities which enter the simulation) can be larger than the number of allocated entities (entity objects consuming memory).

Multilink

A multilink is an attribute of an entity, but the linked entities in that multilink are stored outside the entity. The number of entities in a multilink is variable. The value of an operation on a multilink (such as `sum_over`) is held in an attribute of the entity.

Entity table

The cells of an entity table are in memory outside of entities. Each cell contains accumulators to which entity increments are pushed during the simulation. Each entity contains an `Increment` member for each table which manages the entity's current contribution to the table. Additional members in the entity may be required by a table, for example an identity attribute for the table filter, the lagged value of an attribute, the value of an attribute at the start of the current table increment. An entity table can also contain, for each cell, one or more collections of microdata used to calculate ordinal statistics such as median, Gini coefficient, and percentiles.

Set

The cells of an entity set are in memory outside of entities. Each cell of a set contains a collection of entities which satisfy the conditions to be in the cell. The entity contains a data member for the index of the entity in each set. If the entity set has a filter, the entity will contain an attribute for it.

Event

Each event has an `Event` data member in the associated entity. Pointers to these events are held in an event queue. The event queue of a time-based (interacting population) model can be large because each event of each entity is in the queue. The next occurrence time for an event is recalculated when, as a result of an event, the attributes used in the calculation of the event occurrence time change. Not all events are in the event queue. An event with next occurrence time of infinity is not in the queue, as are events whose future time is explicitly right-censored by model logic using the `censor_event_time` option. Maintaining the event queue can be computationally intensive and account for a significant fraction of model run time. The computational cost of event queue maintenance depends on the number of queue operations and the size of the queue.

[back to topic contents]

Syntax and Use

By default, gathering and reporting of runtime information is disabled. To activate it, include the statement

```
options resource_use = on;
```

in the source code of a model. A natural place to insert this statement is the module `ompp_framework.ompp`.

The resource use report is produced only for sub/replicate/member 0. It is written to the run log when the sub completes.

It is recommended to run with a single sub when using `resource_use`. If the run has multiple subs and they run concurrently the run log file may contain interleaved lines from multiple subs, which could fragment the resource report in the run log.

The `resource_use` option is intended for model development, not model use. With the option on, there might be a slight reduction in run performance and a voluminous report is written to the run log.

If `resource_use` is enabled, the model will write a warning like the following to the log on every model run:

```
Warning : Model built with resource_use = on
```

[back to topic contents]

Illustrated Reference

This subtopic describes each table of the resource use report, using for illustration a run of the GMM model (Genetic Mixing Model). Some model-specific commentary is included to illustrate interpretation of the reports. For context, GMM simulates an interacting population over multiple generations, with genetic inheritance related to cancer. The GMM run for these examples used a starting population of 1 million.

This subtopic contains the following sections. Each section describes one of the tables in the resource use report, and is illustrated with commentary using the GMM run.

- General: General information about the report
- Resource use summary: MB of memory broken out by major categories
- Entity instances: Entity counts and MB
- Entity members: Categorized counts of the data members of the entity
- Entity multilinks: Size of multilinks and MB of the entity
- Entity events: Event frequencies and presence in the event queue for the entity
- Entity sets: Sets of the entity
- Entity tables: Tables of the entity
- Entity member detail: Information on all members of the entity
- Derived tables: Size and MB of derived tables

[back to topic contents]

General

The resource use report consists of a series of sections with one or more tables in each section. Headers in the report demarcate the sections, e.g.

```
************************************
*  Resource Use Detail for Person  *
************************************
```

The report starts with a summary table of memory use. This is followed by a section on resource use for each kind of entity in the model. These entity sections can contain up to 7 tables on resource use for each kind of entity.

GMM has three kinds of entities: `Person`, `Doppleganger`, and `Ticker`, so the GMM report contains three sections with up to 7 tables each. Inapplicable tables are suppressed.

The report concludes with a table on memory use of derived tables.

Memory use in MB are lower bound estimates. Actual memory use depends on implementation details of the C++ standard library and the host operating system.

Each line of the resource report has a timestamp prefix, as do all lines in run logs. These prefixes have been stripped in the example.

For brevity, "run" below means "sub/replicate/member 0 of the run". If the run consists of a single sub as recommended above, the two are the same.

[back to illustrated reference]
[back to topic contents]

Resource use summary

The resource use report begins with a Resource Use Summary section containing two tables. The first table looks like this:

```
+---------------------------+
| Resource Use by Category  |
+-------------------+-------+
| Category          |    MB |
+-------------------+-------+
| Entities          |  1876 |
|   Doppelganger    |   552 |
|   Person          |  1324 |
|   Ticker          |     0 |
| Multilinks        |    10 |
| Events            |    80 |
| Sets              |   196 |
| Tables            |     0 |
|   Entity          |     0 |
|   Derived         |     0 |
| Parameters        |     4 |
|   Fixed           |     0 |
|   Scenario        |     0 |
|   Derived         |     3 |
+-------------------+-------+
| Total             |  2169 |
+-------------------+-------+
```

This table summarizes information found in detailed tables elsewhere in the report (except for parameters, which currently have no detailed section). The first lines give memory use for entities by the kind of entity. Subsequent lines report memory use outside of entities.

In the example, total memory use was a bit over 2 GB, and the bulk of memory use was concentrated in the `Person` entity. Memory use for the `Doppleganger` entity was nevertheless significant at around 0.5 GB. The event queue consumed only 80 MB, and the entity sets used to implement union formation took around 200 MB.

The second table in the Memory Use Summary section looks like this:

```
+--------------------------------+
| Resource Use by Persistence    |
+------------------------+-------+
| Persistence            |    MB |
+------------------------+-------+
| Constant per instance  |     0 |
| Constant per sub       |     4 |
| Variable by pop. size  |  2164 |
+------------------------+-------+
| Total                  |  2169 |
+------------------------+-------+
```

This table regroups memory use to help project the memory required by a run as a function of population size and parallel execution.

Memory use which is independent of population size and shared among simulation threads executing in parallel is reported in row 1.

Memory use which is independent of population size and required by each sub is reported in row 2.

Memory use which varies with population size is reported in row 3. This row is zero for case-based models, by design.

[back to illustrated reference]
[back to topic contents]

Entity instances

This is table 1 of 7 in the Resource Use Detail section for the entity type.

It is a one-row table about instances of the entity.

Here it is for the `Person` entity:

```
+-------------------------------------+
|    Person Instances                 |
+--------------+--------------+-------+
| Activations  | Allocations  |   MB  |
+--------------+--------------+-------+
|     2302315  |     1003104  |  1324 |
+--------------+--------------+-------+
```

Activations is the number of times a `Person` entity entered the simulation. Allocations is the number of times memory for a `Person` entity was allocated. MB is the product of allocations and the size in bytes of a `Person` entity.

Allocations differ from activations because OpenM++ recycles an entity for reuse when it leaves the simulation. So when a `Person` dies and leaves the simulation in GMM the `Person` is placed into a pool and recycled when a new `Person` is born during the simulation.

In the illustration, a bit over half of the `Person` entities were recycled during the run.

Here is the same table for the `Doppleganger` entity:

```
+-------------------------------------+
|    Doppelganger Instances           |
+--------------+--------------+-------+
| Activations  | Allocations  |   MB  |
+--------------+--------------+-------+
|     2302315  |     2302315  |   552 |
+--------------+--------------+-------+
```

Unlike for `Person` entities, the number of Allocations of `Doppleganger` entities is identical to the number of Activations.

In GMM, each `Person` entity has an associated `Doppleganger` entity which duplicates several attributes of the linked `Person` during the simulation. The `Doppleganger` entity has no events or tables of its own and is considerably smaller than the `Person` entity. Unlike the `Person` entity, the `Doppleganger` entity persists after death. That allows the construction of a multi-generational pedigree for the desendants of each `Person`, even if the ancestors of the `Person` are no longer present in the simulation.

[back to illustrated reference]
[back to topic contents]

Entity members

This is table 2 of 7 in the Resource Use Detail section for the entity type.

Here it is for the `Person` entity in the example:

```
+---------------------------+
|    Person Members         |
+-------------------+-------+
| Member            | Count |
+-------------------+-------+
| Attributes        |   201 |
|   Built-in        |     6 |
|   Simple          |    57 |
|   Maintained      |   136 |
|   Link            |     2 |
| Events            |    11 |
| Increments        |     3 |
| Multilink         |     0 |
| Internal          |    60 |
| Array             |     0 |
| Foreign           |     1 |
+-------------------+-------+
| All               |   276 |
+-------------------+-------+
```

This table provides counts of all data members fo the entity, by category. The meaning of each category is given in the following table. Detail on each data member of the entity is reported in the Entity member detail table.

Row	Description
Attributes	All attributes in the entity, including attributes generated by the OpenM++ compiler.
Built-in	Attributes such as `time`, `age`, and `entity_id`.
Simple	Attributes declared in model code with values set explicitly by model code.
Maintained	Automatically maintained attributes such as `duration()`, identity attributes, and operators on multilinks such as `sum_over`.
Link	Links to another entity declared in the model.
Events	Events declared in the model, and possibly the generated event `om_ss_event` for self-scheduling attributes.
Increments	Increments for entity tables, one for each accumulator in each entity table.
Multilink	Multilinks declared in model code.
Internal	Data members (not attributes) created by the OpenM++ compiler to implement the model.
Array	Arrays declared in the model code.
Foreign	Members with a developer-supplied type declared in model code.

In GMM the `Person` entity has 276 data members.

Here's the same table for the `Doppleganger` entity:

```
+----------------------------+
|    Doppelganger Members  |
+-------------------+-------+
| Member            | Count |
+-------------------+-------+
| Attributes        |    44 |
|   Built-in        |     6 |
|   Simple          |    28 |
|   Maintained      |     7 |
|   Link            |     3 |
| Events            |     0 |
| Increments        |     0 |
| Multilink         |     2 |
| Internal          |    10 |
| Array             |     0 |
| Foreign           |     0 |
+-------------------+-------+
| All               |    56 |
+-------------------+-------+
```

The `Doppleganger` entity has fewer members than the `Person` entity. This was a design goal of the GMM architecture. A `Doppleganger` entity contains minimal information echoed from its corresponding `Person` entity. `Doppleganger` entities are deliberately small because they persist after death, which increases memory use as simulation time advances.

From the table, the `Doppleganger` entity has 2 multilinks whereas the `Person` entity has none. That's because the multilinks between parent and children need to persist beyond the lifetime of the parent to generate multi-generational family pedigrees which assess the genetic component of cancer risk of descendants.

[back to illustrated reference]
[back to topic contents]

Entity multilinks

This is table 3 of 7 in the Resource Use Detail section for the entity type.

Here it is for the `Doppelganger` entity in the example:

```
+-----------------------------------------------------------------------+
|    Doppelganger Multilink elements                                    |
+-------------------+--------------+--------------+--------------+-------+
| multilink         |    max size |   entity_id |    avg size |   MB |
+-------------------+--------------+--------------+--------------+-------+
| mlFather_children |          11 |     3241001 |      0.2954 |    5 |
| mlMother_children |           7 |     2794375 |      0.2954 |    5 |
+-------------------+--------------+--------------+--------------+-------+
| All               |             |             |             |   10 |
+-------------------+--------------+--------------+--------------+-------+
```

The meaning of each column is given in the following table.

Column	Description
multilink	The name of the multilink as declared in model code.
max size	The maximum of the multilink size over all entities in the run.
entity_id	An entity_id for which that maximum was attained.
avg size	The average over all entities of the maximum per-entity multilink size.

When `resource_use` is turned on, the maximum multilink size is tracked and the associated `entity_id` recorded and reported in this table to help model developers probe extreme cases or anomalies.

In the example, the maximum number of children of fathers was 11, which occurred for `entity_id` 3241001. The maximum number of children of mothers was 7. This assymetry is an expected consequence of the model design, which acounts for pregnancy, reduced fertility after childbirth, age- and sex-specific union formation and dissolution patterns, and patterns of completed fertility for females.

At first glance the average size of the parent-children multilinks seems small at 0.2954, which is well below replacement fertility. This is an expected consequence of the GMM design for several reasons. First, the starting population of `Person` entities is cross-sectional and those entities have no children. Second, fertility is exogenous during the model's burn-in period when coherent patterns of union formation, union duration, and family structure are being established. Third, all entities exit the run when it ends at year 200, and for many that will occur before lifetime fertility is complete.

[back to illustrated reference]
[back to topic contents]

Entity events

This is table 4 of 7 in the Resource Use Detail section for the entity type.

Here it is for the `Person` entity in the illustrative GMM run:

```
+-----------------------------------------------------------------------------------------------------------+
|    Person Events                                                                                          |
+-----------------------+--------------+--------------+--------------+--------------+--------------+-------+
| event                 |   time calcs |    censored |  occurrences |   per entity | max in queue |    MB |
+-----------------------+--------------+--------------+--------------+--------------+--------------+-------+
| CancerBreastEvent     |    142141567 |    132678956 |       109264 |       0.0475 |        98068 |     3 |
| CancerOvaryEvent      |    142046175 |    140468459 |        13872 |       0.0060 |        16588 |     0 |
| CancerPancreasEvent   |    142059175 |    138987528 |        26872 |       0.0117 |        31477 |     1 |
| CancerProstateEvent   |    142183730 |    129790585 |       151427 |       0.0658 |       120050 |     3 |
| ConceptionEvent       |     66577109 |     48877013 |      1081390 |       0.4697 |       110508 |     3 |
| MortalityEvent        |    142032303 |     91704028 |      2074535 |       0.9011 |       396686 |    12 |
| PregnancyClockEvent   |      6624484 |      3383265 |      3240779 |       1.4076 |         8603 |     0 |
| SexualDebutEvent      |    143503691 |    132811541 |      1471388 |       0.6391 |        87064 |     2 |
| UnionDissolutionEvent |    187014898 |    144258340 |     10000504 |       4.3437 |       257025 |     8 |
| UnionFormationEvent   |     76657431 |     63401111 |     11722623 |       5.0917 |       389374 |    12 |
| om_ss_event           |    431081130 |       457311 |    288052240 |     125.1142 |      1003104 |    32 |
+-----------------------+--------------+--------------+--------------+--------------+--------------+-------+
| All                   |              |              |              |              |              |    80 |
+-----------------------+--------------+--------------+--------------+--------------+--------------+-------+
```

The meaning of each column is given in the following table.

Column	Description
event	The name of the event as given in model code. The event `om_ss_event`, if present, is a generated event which implements all self-scheduling attributes in the entity.

Column	Description
time calcs	The number of calculations of the time of next occurrence of the event.
censored	The number of such calculations which were censored and not inserted into the event queue. Events can be censored either because their next occurrence time is infinity, or by explicit model code in conjunction with the `censor_event_time` option.
occurrences	The number of occurrences of the event during the run.
per entity	The average number of occurrences per entity, calculated as the number of event occurrences divided by the number of activations of the entity type during the run.
max in queue	The maximum number of entities awaiting this event in the event queue during the run.
MB	The number of MB required to store that maximum in the event queue, calculated as the product of the maximum and the size in bytes of an element of the queue datastructure.

In the illustrative GMM run, the average number of union formations is somewhat larger than the number of union dissolutions, which is expected. The frequency of cancer events is as expected, taking into account that the denominator in the table is all `Person` entities, both male and female. The average number of self-scheduling events is high at 288 million. This is probably due to the attribute `self_scheduling_int(age)`, which is listed along with all self-scheduling attributes in the Entity member detail report for `Person`.

[back to illustrated reference]
[back to topic contents]

Entity sets

This is table 5 of 7 in the Resource Use Detail section for the entity type.

Here it is for the `Person` entity in the example:

```
+-----------------------------------------------------------------------------------+
|    Person Sets                                                                    |
+-----------------+-------+----------+----------+--------------+----------+-------+
| set             | rank |   cells | inserts |   per entity | max pop |    MB |
+-----------------+-------+----------+----------+--------------+----------+-------+
| asAllPerson     |    0 |       1 |  2302315 |       1.0000 |  1003104 |    48 |
| asAvailableMen  |    2 |      48 | 14336309 |       6.2269 |   396653 |    19 |
| asAvailableMenV |    3 |      96 | 15484605 |       6.7257 |   396653 |    19 |
+-----------------+-------+----------+----------+--------------+----------+-------+
| All             |      |         |          |              |          |    86 |
+-----------------+-------+----------+----------+--------------+----------+-------+
```

The meaning of each column is given in the following table.

Column	Description
set	The name of the entity set as given in model code.
rank	The number of dimensions in the entity set.
cells	The number of cells in the entity set, calculated as the product of the dimension sizes.
inserts	The number of insert operations on the set during the run, which counts entrances into the set and movements from cell to cell within the set. The number of remove operations is not reported because it is equal to the number of insert operations over the course of a run.
per entity	The number of inserts per entity, calculated as the number of inserts divided by the number of activations of the entity type during the run.
max pop	The maximum count of entities in the set during the run.
MB	The number of MB required to store that maximum count in the entity set, calculated as the product of the maximum count and the size in bytes of an element of the set datastructure.

In the example, the set `asAllPerson` has a single cell and contains all `Person` entities. This set is used to communicate the passage of integer time (year) from the single `Ticker` entity to all `Person` entities during the run. The maximum population in this set is slightly higher than the starting

population of 1 million, perhaps due to Monte Carlo variation during the burn-in phase of the run.

The other two entity sets help implement a female choice model of union formation. The churning in these sets is a consequence of changes in union status ('single' is a filter condition of these sets) and age group (a dimension of these sets).

[back to illustrated reference]
[back to topic contents]

Entity tables

This is table 6 of 7 in the Resource Use Detail section for the entity type.

Here it is for the `Person` entity in the GMM run:

```
+-----------------------------------------------------------------------------------------------------------------------+
|    Person Tables                                                                                                      |
+---------------------------------------+-------+----------+-------+-------+-------+----------+--------------+-------+
| table                                 | rank |   cells | accum | measr | colls | incrmnts |   per entity |    MB |
+---------------------------------------+-------+----------+-------+-------+-------+----------+--------------+-------+
| Boadicea_RR_cross                     |    2 |     256 |     3 |     3 |     0 |   154304 |       0.0670 |     0 |
| IM_RiskEvaluationDistributionDetailed |    2 |     150 |     1 |     1 |     0 |   154304 |       0.0670 |     0 |
| PersonEvents                          |    2 |    2222 |     1 |     1 |     0 | 168607252 |      73.2338 |     0 |
+---------------------------------------+-------+----------+-------+-------+-------+----------+--------------+-------+
| All                                   |      |         |      |      |      |         |              |     0 |
+---------------------------------------+-------+----------+-------+-------+-------+----------+--------------+-------+
```

Entity tables suppressed at run-time are not included in this report. If any are suppressed at run-time, a message with the number of run-time suppressed tables follows immediately after the table.

The meaning of each column is given in the following table.

Column	Description
table	The name of the entity table as given in model code.
rank	The number of dimensions in the table.
cells	The number of cells in the table, calculated as the product of the dimension sizes, including margins if present.
accum	The number of accumulators in the table.
measr	The number of measures in the table.
colls	The number of collections in the table.
incrmnts	The number of increments pushed to the table during the run.
per entity	The number of increments per entity, calculated as the number of increments divided by the number of activations of the entity type during the run.
MB	The number of MB required by the cells of the table, calculated as the product of the number of cells, the number of accumulators, and the size in bytes of a `double`.

GMM contains many tables, but most are suppressed in the production variant of the model using `tables_retain` statements. The `PersonEvents` table was retained, but given the high volume of increments shown in the table above (168m) and its diagnostic nature it is a candidate for suppression in the production variant of GMM.

[back to illustrated reference]
[back to topic contents]

Entity member detail

This is table 7 of 7 in the Resource Use Detail section for the entity type.

This table contains one row for each entity member, and so can have many rows. Here is an extract for the `Person` entity for the GMM example.

```
+-----------------------------------------------+-------+
|    Person Members (detail)                          |
+-----------------------------------------------+-------+
| member                                      | bytes |
+-----------------------------------------------+-------+
| Attributes:                                 |       |
|   Built-in:                                 |       |
|     age                                     |     8 |
|     entity_id                               |     4 |
|     time                                    |     8 |
|   Simple:                                   |       |
|     alive                                   |     1 |
|     union_status                            |     1 |
|     year                                    |     1 |
|   Maintained:                               |       |
|     age_group                               |     1 |
|     can_conceive                            |     1 |
...
| Array:                                      |       |
| Foreign:                                    |       |
|     breast_cancer_hazard                    |    24 |
+-----------------------------------------------+-------+
| Sum of member bytes                         |  1170 |
| Bytes per entity                            |  1320 |
| Storage efficiency (%)                      |  88.6 |
+-----------------------------------------------+-------+
```

The rows of this table are grouped like the Entity members summary table. Each member has a row with the member name followed by its size in bytes. Some members generated by the compiler have non-intuitive names which are replaced by more easily interpreted text in the report.

There are three summary rows at the end of the table: the sum of member bytes, the actual size of the entity in bytes, and the implied storage efficiency. For more on member packing and storage efficiency see Entity Member Packing.

For GMM, the Entity Member Detail table has around 300 rows. For the full version see the Appendix of this topic.

The following table extracts selected rows from the full table and adds some remarks to aid interpretation.

member	bytes	remarks
Attributes:		
Built-in:		
`age`	8	Automatically maintained age of the entity, of type `Time`.
`case_seed`	8	Not sure why this member is present in a time-based model like GMM.
`censor_time`	8	Created and maintained if the `censor_event_time` option is activated.
`entity_id`	4	Automatically assigned unique numeric identifier of the entity.
`events`	2	An automatically maintained counter of events undergone by the entity.
`time`	8	Automatically maintained time for the entity, of type `Time`.
Simple:		
`alive`	1	Declared in model as type `bool`. Note that a `bool` takes one byte of storage, not one bit.
`union_status`	1	An attribute of type `UNION_STATUS` declared in the model. `UNION_STATUS` is a classification with 3 levels, and is stored efficiently in a single byte.
`year`	1	Declared in model as type `YEAR`, which is a range declared as `{0,200}`. It is stored efficienty in a single byte. The value of `year` is maintained by the `Ticker` entity which updates it for all `Person` entities by iterating the `asAllPersons` entity set when the `TickEvent` occurs.
Maintained:		
`can_conceive`	1	An identity attribute declared in model code by `logical can_conceive = fertile && stable_union && eligible_union`
`om_aia_0`	1	An identity attribute created by the compiler to implement other attributes.

member	bytes	remarks
`om_asAvailableMen_filter`	1	An identity attribute created by the compiler to implement the filter of the entity set `asAvailableMen`.
`entrances(union_status,US_SINGLE)`	2	A maintained attribute declared implicitly in model code.
`self_scheduling_int(age)`	2	A maintained attribute declared implicitly in model code. This is a self-scheduling attribute, so the compiler will also create the special self-scheduling event `om_ss_event` to implement it and any other self-scheduling attributes in the entity.
`trigger_changes(year)`	1	A self-scheduling attribute declared implicitly in model code. Used in table filters to create snapshot-style tables.
Link:		
`lDoppelganger`	8	A `link` attribute declared in model code.
`lPartner`	8	Another `link` attribute.
Events:		
`ConceptionEvent`	24	The member used to manage the `ConceptionEvent` event declared in model code.
`MortalityEvent`	24	The member for another event declared in the model.
`om_ss_event`	24	The generated event which schedules and implements all self-scheduling attributes in the entity. This event is not present if the entity has no self-scheduling attributes.
Increments:		
`Boadicea_RR_cross increment`	16	The member managing the current increment of the `Person` for the table `Boadicea_RR_cross`.
`PersonEvents increment`	16	Another increment, this one for the `PersonEvents` table.
Multilink:		The `Person` entity has no multilinks. Multilinks for children are held in the parallel `Doppleganger` entity.
Internal:		
`PersonEvents (inevent) event_count`	4	Supports tabulation of `event_count` using the `event` keyword in the `PersonEvents` table.
`asAvailableMen (current cell)`	4	The cell in the entity set `asAvailableMen` currently occupied by the `Person`, provided it meets the filter condition of `asAvailableMen`.
`event_count (lagged)`	4	Holds the lagged value of the attribute `event_count`. Lagged values are required by some table operators.
`event_count (counter at lagged)`	8	Used to detect changes in the lagged value of `event_count`.
`om_self_scheduling_int_FOR_age (next time)`	8	Contains the next scheduled time of the self-scheduling attribute `self_scheduling_int(age)`.
Array:		There are no array members of `Person` in the GMM model.
Foreign:		
`breast_cancer_hazard`	24	Declared in model code as type `std::vector<double>` for reasons explained by the model developer in model source code comments.

[back to illustrated reference]
[back to topic contents]

Derived tables

This is the final table of the resource use report.

Here it is for the GMM run:

```
+-----------------------------------------------------------------+
|     Derived Tables                                              |
+-------------------------------+-------+----------+-------+-------+
| derived table                 | rank |    cells | measr |    MB |
+-------------------------------+-------+----------+-------+-------+
| IM_GMM_KeyInputs              |    1 |        7 |     1 |     0 |
| IM_IncidenceRR                |    3 |    24480 |     1 |     0 |
| IM_MajorGeneBirthDistribution |    1 |        6 |     1 |     0 |
| IM_OncogenesisRR              |    3 |    24480 |     1 |     0 |
| IM_PolygeneBirthDistribution  |    1 |       51 |     1 |     0 |
| IM_RiskEvaluationAge          |    0 |        1 |     1 |     0 |
+-------------------------------+-------+----------+-------+-------+
| All                           |      |          |       |     0 |
+-------------------------------+-------+----------+-------+-------+
```

Derived tables suppressed at run-time are not included in this report. If there are any such, a message giving the number of derived tables suppressed at run-time follows immediately after the table.

The meaning of each column is given in the following table.

Column	Description
derived table	The name of the derived table as given in model code.
rank	The number of dimensions in the table.
cells	The number of cells in the table, calculated as the product of the dimension sizes, including margins if present.
measr	The number of measures in the table.
MB	The number of MB required by the cells of the table, calculated as the product of the number of cells, the number of measures, and the size in bytes of a `double`.

GMM contains other derived tables which are suppressed in the production variant of the model using `tables_retain` statements. Retained tables in GMM include those required to feed the downstream `OncoSim` model which simulates the health system consequences of genetic risk-based screening simulated in a GMM run. The GMM model code prefixes those tables with `IM_` to distinguish them.

[back to illustrated reference]
[back to topic contents]

Appendix

For reference, here is the complete report of the GMM run used in the illustrations.

```
Resource Use Report - Begin (for sub/member/replicate 0)

 **************************
 *  Resource Use Summary  *
 **************************

  +---------------------------+
  | Resource Use by Category  |
  +-------------------+-------+
  | Category          |    MB |
  +-------------------+-------+
  | Entities          |  1876 |
  |   Doppelganger    |   552 |
  |   Person          |  1324 |
  |   Ticker          |     0 |
  | Multilinks        |    10 |
  | Events            |    80 |
  | Sets              |   196 |
  | Tables            |     0 |
  |   Entity          |     0 |
  |   Derived         |     0 |
  | Parameters        |     4 |
  |   Fixed           |     0 |
  |   Scenario        |     0 |
  |   Derived         |     3 |
  +-------------------+-------+
  | Total             |  2169 |
  +-------------------+-------+
```

```
  +--------------------------------+
  | Resource Use by Persistence    |
  +------------------------+-------+
  | Persistence            |    MB |
  +------------------------+-------+
  | Constant per instance  |     0 |
  | Constant per sub       |     4 |
  | Variable by pop. size  |  2164 |
  +------------------------+-------+
  | Total                  |  2169 |
  +------------------------+-------+

*****************************************
*  Resource Use Detail for Doppelganger  *
*****************************************

  +-------------------------------------+
  |     Doppelganger Instances          |
  +--------------+--------------+-------+
  | Activations  | Allocations  |    MB |
  +--------------+--------------+-------+
  |      2302315 |      2302315 |   552 |
  +--------------+--------------+-------+
  Note: MB does not include storage of elements of multilinks

  +----------------------------+
  |     Doppelganger Members   |
  +--------------------+-------+
  | Member             | Count |
  +--------------------+-------+
  | Attributes         |    44 |
  |   Built-in         |     6 |
  |   Simple           |    28 |
  |   Maintained       |     7 |
  |   Link             |     3 |
  | Events             |     0 |
  | Increments         |     0 |
  | Multilink          |     2 |
  | Internal           |    10 |
  | Array              |     0 |
  | Foreign            |     0 |
  +--------------------+-------+
  | All                |    56 |
  +--------------------+-------+

  +-----------------------------------------------------------------------+
  |     Doppelganger Multilink elements                                   |
  +-------------------+--------------+--------------+--------------+-------+
  | multilink         |     max size |    entity_id |     avg size |    MB |
  +-------------------+--------------+--------------+--------------+-------+
  | mlFather_children |           11 |      3241001 |       0.2954 |     5 |
  | mlMother_children |            7 |      2794375 |       0.2954 |     5 |
  +-------------------+--------------+--------------+--------------+-------+
  | All               |              |              |              |    10 |
  +-------------------+--------------+--------------+--------------+-------+
  Note: MB does not include entity members

  +-----------------------------------------------------------------------------------+
  |     Doppelganger Sets                                                             |
  +-------------------+-------+----------+----------+--------------+----------+-------+
  | set               |  rank |    cells |  inserts |   per entity |  max pop |    MB |
  +-------------------+-------+----------+----------+--------------+----------+-------+
  | asAllDoppelganger |     0 |        1 |  2302315 |       1.0000 |  2302315 |   110 |
  +-------------------+-------+----------+----------+--------------+----------+-------+
  | All               |       |          |          |              |          |   110 |
  +-------------------+-------+----------+----------+--------------+----------+-------+
  Note: MB does not include entity members

  +-----------------------------------------------+
  |     Doppelganger Members (detail)             |
  +---------------------------------------+-------+
  | member                                | bytes |
  +---------------------------------------+-------+
  | Attributes:                           |       |
  |   Built-in:                           |       |
  |     age                               |     8 |
  |     case_seed                         |     8 |
  |     censor_time                       |     8 |
  |     entity_id                         |     4 |
  |     events                            |     2 |
  |     time                              |     8 |
  |   Simple:                             |       |
  |     boadicea_risk_factor_code         |     4 |
  |     children                          |     4 |
  |     endogenous_grandparent_count      |     1 |
  |     enhanced_screening_reason         |     1 |
  |     g1_atm_mutation                   |     1 |
  |     g1_brca1_mutation                 |     1 |
```

```
|     g1_brca1_mutation                  |     1 |
|     g1_brca2_mutation                  |     1 |
|     g1_chek2_mutation                  |     1 |
|     g1_palb2_mutation                  |     1 |
|     g2_atm_mutation                    |     1 |
|     g2_brca1_mutation                  |     1 |
|     g2_brca2_mutation                  |     1 |
|     g2_chek2_mutation                  |     1 |
|     g2_palb2_mutation                  |     1 |
|     grandparent_count                  |     1 |
|     major_genotype                     |     1 |
|     one_column                         |     1 |
|     polygene_bin                       |     1 |
|     polygenic_known                    |     4 |
|     polygenic_unknown                  |     4 |
|     sex                                |     1 |
|     year_born                          |     2 |
|     year_cancer_breast                 |     2 |
|     year_cancer_ovary                  |     2 |
|     year_cancer_pancreas               |     2 |
|     year_cancer_prostate               |     2 |
|     year_died                          |     2 |
|     year_start_enhanced_screening      |     2 |
|   Maintained:                          |       |
|     any_mutation                       |     1 |
|     atm_mutation                       |     1 |
|     brca1_mutation                     |     1 |
|     brca2_mutation                     |     1 |
|     chek2_mutation                     |     1 |
|     palb2_mutation                     |     1 |
|     polygenic_total                    |     4 |
|   Link:                                |       |
|     lFather                            |     8 |
|     lMother                            |     8 |
|     lPerson                            |     8 |
| Events:                                |       |
| Increments:                            |       |
| Multilink:                             |       |
|     mlFather_children                  |    24 |
|     mlMother_children                  |    24 |
| Internal:                              |       |
|     MajorGenotype1 (in) any_mutation   |     1 |
|     MajorGenotype1 (in) atm_mutation   |     1 |
|     MajorGenotype1 (in) brca1_mutation |     1 |
|     MajorGenotype1 (in) brca2_mutation |     1 |
|     MajorGenotype1 (in) chek2_mutation |     1 |
|     MajorGenotype1 (in) palb2_mutation |     1 |
|     Polygene1 (in) polygenic_known     |     4 |
|     Polygene1 (in) polygenic_total     |     4 |
|     Polygene1 (in) polygenic_unknown   |     4 |
|     asAllDoppelganger (current cell)   |     4 |
| Array:                                 |       |
| Foreign:                               |       |
+----------------------------------------+-------+
| Sum of member bytes                    |   189 |
| Bytes per entity                       |   240 |
| Storage efficiency (%)                 |  78.8 |
+----------------------------------------+-------+

************************************
*  Resource Use Detail for Person  *
************************************

  +-------------------------------------+
  |     Person Instances                |
  +--------------+--------------+-------+
  | Activations  | Allocations  |    MB |
  +--------------+--------------+-------+
  |      2302315 |      1003104 |  1324 |
  +--------------+--------------+-------+
  Note: MB does not include storage of elements of foreign objects

  +----------------------------+
  |     Person Members         |
  +--------------------+-------+
  | Member             | Count |
  +--------------------+-------+
  | Attributes         |   201 |
  |   Built-in         |     6 |
  |   Simple           |    57 |
  |   Maintained       |   136 |
  |   Link             |     2 |
  | Events             |    11 |
  | Increments         |     3 |
  | Multilink          |     0 |
  | Internal           |    60 |
  | Array              |     0 |
  | Foreign            |     1 |
```

```
+---------------------+-------+
| All                 |   276 |
+---------------------+-------+

+-----------------------------------------------------------------------------------------------------------+
|     Person Events                                                                                         |
+-----------------------+--------------+--------------+--------------+--------------+--------------+-------+
| event                 |   time calcs |     censored |  occurrences |   per entity | max in queue |    MB |
+-----------------------+--------------+--------------+--------------+--------------+--------------+-------+
| CancerBreastEvent     |    142141567 |    132678956 |       109264 |       0.0475 |        98068 |     3 |
| CancerOvaryEvent      |    142046175 |    140468459 |        13872 |       0.0060 |        16588 |     0 |
| CancerPancreasEvent   |    142059175 |    138987528 |        26872 |       0.0117 |        31477 |     1 |
| CancerProstateEvent   |    142183730 |    129790585 |       151427 |       0.0658 |       120050 |     3 |
| ConceptionEvent       |     66577109 |     48877013 |      1081390 |       0.4697 |       110508 |     3 |
| MortalityEvent        |    142032303 |     91704028 |      2074535 |       0.9011 |       396686 |    12 |
| PregnancyClockEvent   |      6624484 |      3383265 |      3240779 |       1.4076 |         8603 |     0 |
| SexualDebutEvent      |    143503691 |    132811541 |      1471388 |       0.6391 |        87064 |     2 |
| UnionDissolutionEvent |    187014898 |    144258340 |     10000504 |       4.3437 |       257025 |     8 |
| UnionFormationEvent   |     76657431 |     63401111 |     11722623 |       5.0917 |       389374 |    12 |
| om_ss_event           |    431081130 |       457311 |    288052240 |     125.1142 |      1003104 |    32 |
+-----------------------+--------------+--------------+--------------+--------------+--------------+-------+
| All                   |              |              |              |              |              |    80 |
+-----------------------+--------------+--------------+--------------+--------------+--------------+-------+
Note: MB does not include entity members

+-------------------------------------------------------------------------------------+
|     Person Sets                                                                     |
+-----------------+-------+----------+----------+--------------+----------+-------+
| set             |  rank |    cells |  inserts |   per entity |  max pop |    MB |
+-----------------+-------+----------+----------+--------------+----------+-------+
| asAllPerson     |     0 |        1 |  2302315 |       1.0000 |  1003104 |    48 |
| asAvailableMen  |     2 |       48 | 14336309 |       6.2269 |   396653 |    19 |
| asAvailableMenV |     3 |       96 | 15484605 |       6.7257 |   396653 |    19 |
+-----------------+-------+----------+----------+--------------+----------+-------+
| All             |       |          |          |              |          |    86 |
+-----------------+-------+----------+----------+--------------+----------+-------+
Note: MB does not include entity members

+-------------------------------------------------------------------------------------------------------------------------+
|     Person Tables                                                                                                       |
+---------------------------------------+-------+----------+-------+-------+-------+----------+--------------+-------+
| table                                 |  rank |    cells | accum | measr | colls | incrmnts |   per entity |    MB |
+---------------------------------------+-------+----------+-------+-------+-------+----------+--------------+-------+
| Boadicea_RR_cross                     |     2 |      256 |     3 |     3 |     0 |   154304 |       0.0670 |     0 |
| IM_RiskEvaluationDistributionDetailed |     2 |      150 |     1 |     1 |     0 |   154304 |       0.0670 |     0 |
| PersonEvents                          |     2 |     2222 |     1 |     1 |     0 | 168607252 |      73.2338 |     0 |
+---------------------------------------+-------+----------+-------+-------+-------+----------+--------------+-------+
| All                                   |       |          |       |       |       |          |              |     0 |
+---------------------------------------+-------+----------+-------+-------+-------+----------+--------------+-------+
Note: MB does not include entity members

+-----------------------------------------------------------------------------------------+
|     Person Members (detail)                                                             |
+-----------------------------------------------------------------------------------------+-------+
| member                                                                                  | bytes |
+-----------------------------------------------------------------------------------------+-------+
| Attributes:                                                                             |       |
|   Built-in:                                                                             |       |
|     age                                                                                 |     8 |
|     case_seed                                                                           |     8 |
|     censor_time                                                                         |     8 |
|     entity_id                                                                           |     4 |
|     events                                                                              |     2 |
|     time                                                                                |     8 |
|   Simple:                                                                               |       |
|     PRS_top_5                                                                           |     1 |
|     activity_level                                                                      |     1 |
|     actual_RR                                                                           |     8 |
|     actual_risk                                                                         |     8 |
|     alive                                                                               |     1 |
|     base_risk                                                                           |     8 |
|     boadicea_CPU                                                                        |     8 |
|     boadicea_RR                                                                         |     8 |
|     boadicea_evaluations                                                                |     4 |
|     boadicea_risk                                                                       |     8 |
|     cancer_breast                                                                       |     1 |
|     cancer_ovary                                                                        |     1 |
|     cancer_pancreas                                                                     |     1 |
|     cancer_prostate                                                                     |     1 |
|     children_born                                                                       |     4 |
|     eligible_union                                                                      |     1 |
|     endogenous_grandparent_count                                                        |     1 |
|     endogenous_parent_count                                                             |     1 |
|     enhanced_screening                                                                  |     1 |
|     enhanced_screening_reason                                                           |     1 |
|     entry_year                                                                          |     1 |
|     event_count                                                                         |     4 |
|     event_type                                                                          |     1 |
```

```
|    generation                                          |    1 |
|    grandparent_count                                       |    1 |
|    has_any_mutation                                        |    1 |
|    in_union_ey                                         |    1 |
|    in_union_ybey                                        |    1 |
|    is_in_starting_population                               |    1 |
|    major_genotype                                       |    1 |
|    match_failures_forbidden                                |    4 |
|    match_failures_none                                     |    4 |
|    match_successes                                         |    4 |
|    mother_cancer                                        |    1 |
|    one_column                                           |    1 |
|    parent_count                                         |    1 |
|    polygene_bin                                         |    1 |
|    post_partum                                          |    1 |
|    pregnant                                            |    1 |
|    ps                                                |    1 |
|    risk_assessments                                        |    4 |
|    risk_factor_alcohol                                     |    1 |
|    risk_factor_bmi                                         |    1 |
|    risk_factor_first_b                                     |    1 |
|    risk_factor_mammo_density                                       |    1 |
|    risk_factor_menarche                                    |    1 |
|    risk_factor_menopause                                   |    1 |
|    risk_factor_mht                                         |    1 |
|    risk_factor_oc                                          |    1 |
|    risk_factor_parity                                      |    1 |
|    sex                                               |    1 |
|    simulate_risk_factors                                   |    1 |
|    time_pregnancy_clock                                    |    8 |
|    twin_pregnancy                                          |    1 |
|    undergone_debut                                         |    1 |
|    union_status                                            |    1 |
|    year                                              |    1 |
|  Maintained:                                           |      |
|    age_group                                               |    1 |
|    age_group1                                              |    1 |
|    age_group_15                                            |    1 |
|    age_group_for_debut                                     |    4 |
|    ar_er                                             |    8 |
|    boadicea_horizon                                        |    4 |
|    can_conceive                                            |    1 |
|    children0                                           |    1 |
|    children1                                           |    1 |
|    children2                                           |    1 |
|    children3                                           |    1 |
|    children4                                           |    1 |
|    children4plus                                           |    1 |
|    children5                                           |    1 |
|    children6                                           |    1 |
|    children7                                           |    1 |
|    children8plus                                           |    1 |
|    delta_partners_ybey                                     |    4 |
|    diff_RR                                             |    8 |
|    diff_RR_abs                                             |    8 |
|    dissolutions                                        |    4 |
|    ever_had_sex                                            |    1 |
|    fertile                                           |    1 |
|    fertile_age                                         |    1 |
|    genotype_group                                          |    1 |
|    in_casual_union                                         |    1 |
|    in_scope_fertility                                      |    1 |
|    in_stable_union                                         |    1 |
|    in_union                                            |    1 |
|    in_utero                                            |    1 |
|    incidenceRR50                                           |    8 |
|    incidence_rr                                            |    8 |
|    integer_age                                             |    4 |
|    is_ey                                             |    1 |
|    multiple_fathers                                        |    1 |
|    my_year                                             |    1 |
|    active_spell_delta(year_spell,true,partners)                          |    2 |
|    om_aia_0                                                |    1 |
|    om_aia_1                                                |    1 |
|    om_aia_10                                               |    1 |
|    om_aia_11                                               |    1 |
|    om_aia_12                                               |    1 |
|    om_aia_13                                               |    1 |
|    om_aia_14                                               |    1 |
|    om_aia_15                                               |    1 |
|    om_aia_16                                               |    1 |
|    om_aia_17                                               |    1 |
|    om_aia_18                                               |    1 |
|    om_aia_19                                               |    1 |
|    om_aia_2                                                |    1 |
|    om_aia_20                                               |    1 |
|    om_aia_21                                               |    1 |
```

```
|    om_aia_22                                                    |    1 |
|    om_aia_23                                                    |    1 |
|    om_aia_24                                                    |    1 |
|    om_aia_25                                                    |    1 |
|    om_aia_26                                                    |    1 |
|    om_aia_27                                                    |    1 |
|    om_aia_28                                                    |    1 |
|    om_aia_29                                                    |    4 |
|    om_aia_3                                                     |    1 |
|    om_aia_30                                                    |    1 |
|    om_aia_31                                                    |    1 |
|    om_aia_32                                                    |    1 |
|    om_aia_4                                                     |    1 |
|    om_aia_5                                                     |    1 |
|    om_aia_6                                                     |    1 |
|    om_aia_7                                                     |    1 |
|    om_aia_8                                                     |    1 |
|    om_aia_9                                                     |    1 |
|    om_asAvailableMenV_filter                                    |    1 |
|    om_asAvailableMen_filter                                     |    1 |
|    completed_spell_delta(year_spell,true,partners)              |    2 |
|    duration()                                                   |    8 |
|    duration(activity_level,AL_0)                                |    8 |
|    duration(activity_level,AL_1)                                |    8 |
|    duration(activity_level,AL_2)                                |    8 |
|    duration(activity_level,AL_3)                                |    8 |
|    duration(boadicea_evaluations,1)                             |    8 |
|    duration(can_conceive,true)                                  |    8 |
|    duration(cancer_breast,true)                                 |    8 |
|    duration(enhanced_screening,true)                            |    8 |
|    duration(undergone_debut,true)                               |    8 |
|    entrances(cancer_breast,true)                                |    2 |
|    entrances(cancer_ovary,true)                                 |    2 |
|    entrances(cancer_pancreas,true)                              |    2 |
|    entrances(cancer_prostate,true)                              |    2 |
|    entrances(in_stable_union,true)                              |    2 |
|    entrances(union_status,US_CASUAL)                            |    2 |
|    entrances(union_status,US_SINGLE)                            |    2 |
|    entrances(union_status,US_STABLE)                            |    2 |
|    self_scheduling_int(age)                                     |    2 |
|    self_scheduling_int(duration(boadicea_evaluations,1))        |    2 |
|    split(activity_level,ACTIVITY_3)                             |    1 |
|    split(actual_RR,PARTITION_RR)                                |    1 |
|    split(boadicea_RR,IM_PARTITION_RR)                           |    1 |
|    split(boadicea_RR,PARTITION_RR)                              |    1 |
|    split(diff_RR,DIFF_RR)                                       |    1 |
|    split(integer_age,AGE_GROUP15_1549)                          |    1 |
|    split(integer_age,AGE_GROUP5_1549)                           |    1 |
|    split(integer_age,AGE_GROUP5_1559)                           |    1 |
|    split(integer_age,GROUPE_AGE2_1070)                          |    1 |
|    split(integer_age,P_AGE_FOR_SEXUAL_DEBUT)                    |    1 |
|    split(partners_ybey,PARTNERS_5)                              |    1 |
|    trigger_changes(boadicea_evaluations)                        |    1 |
|    trigger_changes(cancer_breast)                               |    1 |
|    trigger_changes(year)                                        |    1 |
|    trigger_entrances(boadicea_evaluations,1)                    |    1 |
|    trigger_entrances(integer_age,0)                             |    1 |
|    trigger_exits(children_born,0)                               |    1 |
|    trigger_exits(children_born,1)                               |    1 |
|    trigger_exits(integer_age,39)                                |    1 |
|    undergone_change(partners)                                   |    1 |
|    undergone_change(union_status)                               |    1 |
|    undergone_entrance(pregnant,true)                            |    1 |
|    undergone_entrance(risk_assessments,1)                       |    1 |
|    value_at_first_entrance(pregnant,true,union_number)          |    4 |
|    value_at_first_entrance(risk_assessments,1,cancer_breast)    |    1 |
|    value_at_latest_entrance(om_aia_17,true,partners)            |    4 |
|    value_at_latest_entrance(pregnant,true,union_number)         |    4 |
|    parity                                                       |    1 |
|    partners                                                     |    4 |
|    partners_casual                                              |    4 |
|    partners_stable                                              |    4 |
|    partners_ybey                                                |    4 |
|    polygenic_total                                              |    4 |
|    possible_proband                                             |    1 |
|    range_age                                                    |    1 |
|    range_age_80                                                 |    1 |
|    stable_union                                                 |    1 |
|    union_number                                                 |    4 |
|    union_number_at_first_conception                             |    4 |
|    union_number_at_latest_conception                            |    4 |
|    virgin                                                       |    1 |
|    year_flash                                                   |    1 |
|    year_spell                                                   |    1 |
|  Link:                                                          |      |
|    lDoppelganger                                                |    8 |
|    lPartner                                                     |    8 |
| Events:                                                         |      |
```

```
|     CancerBreastEvent                                                    |    24 |
|     CancerOvaryEvent                                                     |    24 |
|     CancerPancreasEvent                                                  |    24 |
|     CancerProstateEvent                                                  |    24 |
|     ConceptionEvent                                                      |    24 |
|     MortalityEvent                                                       |    24 |
|     PregnancyClockEvent                                                  |    24 |
|     SexualDebutEvent                                                     |    24 |
|     UnionDissolutionEvent                                                |    24 |
|     UnionFormationEvent                                                  |    24 |
|     om_ss_event                                                          |    24 |
| Increments:                                                              |       |
|     Boadicea_RR_cross increment                                          |    16 |
|     IM_RiskEvaluationDistributionDetailed increment                      |    16 |
|     PersonEvents increment                                               |    16 |
| Multilink:                                                               |       |
| Internal:                                                                |       |
|     ALT_ActivityLevel (in) om_duration                                   |     8 |
|     ALT_ActivityLevel (in) om_duration_FOR_activity_level_X_AL_0         |     8 |
|     ALT_ActivityLevel (in) om_duration_FOR_activity_level_X_AL_1         |     8 |
|     ALT_ActivityLevel (in) om_duration_FOR_activity_level_X_AL_2         |     8 |
|     ALT_ActivityLevel (in) om_duration_FOR_activity_level_X_AL_3         |     8 |
|     BirthRate (in) children_born                                         |     4 |
|     BirthRate (in) om_duration                                           |     8 |
|     BreastCancerRates (in) om_duration                                   |     8 |
|     BreastCancerRates (in) om_duration_FOR_cancer_breast_X_true          |     8 |
|     BreastCancerRates (in) om_entrances_FOR_cancer_breast_X_true         |     2 |
|     CanConceive (in) om_duration                                         |     8 |
|     CanConceive (in) om_duration_FOR_can_conceive_X_true                 |     8 |
|     CancerRates (in) om_duration                                         |     8 |
|     CancerRates (in) om_duration_FOR_cancer_breast_X_true                |     8 |
|     CancerRates (in) om_entrances_FOR_cancer_breast_X_true               |     2 |
|     CancerRates (in) om_entrances_FOR_cancer_ovary_X_true                |     2 |
|     CancerRates (in) om_entrances_FOR_cancer_pancreas_X_true             |     2 |
|     CancerRates (in) om_entrances_FOR_cancer_prostate_X_true             |     2 |
|     FertilityByAge (in) children_born                                    |     4 |
|     FertilityByAge (in) om_duration                                      |     8 |
|     InUtero (in) om_duration                                             |     8 |
|     MatchReport (in) match_failures_forbidden                            |     4 |
|     MatchReport (in) match_failures_none                                 |     4 |
|     MatchReport (in) match_successes                                     |     4 |
|     PYAgeZero (in) om_duration                                           |     8 |
|     PersonEvents (inevent) event_count                                   |     4 |
|     Polygene1a (in) polygenic_total                                      |     4 |
|     PopulationByYear (in) om_duration                                    |     8 |
|     SDT_SexualDebut (in) om_duration                                     |     8 |
|     SDT_SexualDebut (in) om_duration_FOR_undergone_debut_X_true          |     8 |
|     Screening1 (in) om_duration                                          |     8 |
|     Screening1 (in) om_duration_FOR_enhanced_screening_X_true            |     8 |
|     Screening1 (in) risk_assessments                                     |     4 |
|     Screening2 (in) cancer_breast                                        |     1 |
|     Screening2 (in) om_duration                                          |     8 |
|     Screening3 (in) cancer_breast                                        |     1 |
|     Screening3 (in) om_duration                                          |     8 |
|     Screening4 (in) cancer_breast                                        |     1 |
|     Screening5 (in) cancer_breast                                        |     1 |
|     SimulatedEverHadSex (in) ever_had_sex                                |     1 |
|     SimulatedProportionStable (in) in_stable_union                       |     1 |
|     X4_Observed_RR (in) cancer_breast                                    |     1 |
|     X4_Observed_RR (in) om_duration                                      |     8 |
|     X5_Difference_RR (in) diff_RR_abs                                    |     8 |
|     X5_Difference_RR (in) entity_id                                      |     4 |
|     asAllPerson (current cell)                                           |     4 |
|     asAvailableMenV (current cell)                                       |     4 |
|     asAvailableMen (current cell)                                        |     4 |
|     event_count (lagged)                                                 |     4 |
|     event_count (counter at lagged)                                      |     8 |
|     om_self_scheduling_int_FOR_age (next time)                           |     8 |
|     om_self_scheduling_int_FOR_om_duration_FOR_boadicea_evaluations_X_1 (next time) |     8 |
|     om_trigger_changes_FOR_boadicea_evaluations (next time)              |     8 |
|     om_trigger_changes_FOR_cancer_breast (next time)                     |     8 |
|     om_trigger_changes_FOR_year (next time)                              |     8 |
|     om_trigger_entrances_FOR_boadicea_evaluations_X_1 (next time)        |     8 |
|     om_trigger_entrances_FOR_integer_age_X_0 (next time)                 |     8 |
|     om_trigger_exits_FOR_children_born_X_0 (next time)                   |     8 |
|     om_trigger_exits_FOR_children_born_X_1 (next time)                   |     8 |
|     om_trigger_exits_FOR_integer_age_X_39 (next time)                    |     8 |
| Array:                                                                   |       |
| Foreign:                                                                 |       |
|     breast_cancer_hazard                                                 |    24 |
+-------------------------------------------------------------------------+-------+
| Sum of member bytes                                                      | 1170 |
| Bytes per entity                                                         | 1320 |
| Storage efficiency (%)                                                   | 88.6 |
+-------------------------------------------------------------------------+-------+

************************************
*  Resource Use Detail for Ticker  *
```

```
  Resource Use Detail for Ticker
************************************

  +-------------------------------------+
  |     Ticker Instances                |
  +--------------+--------------+-------+
  | Activations  | Allocations  |    MB |
  +--------------+--------------+-------+
  |            1 |            1 |     0 |
  +--------------+--------------+-------+

  +----------------------------+
  |     Ticker Members         |
  +---------------------+-------+
  | Member              | Count |
  +---------------------+-------+
  | Attributes          |    15 |
  |   Built-in          |     6 |
  |   Simple            |     8 |
  |   Maintained        |     1 |
  |   Link              |     0 |
  | Events              |     2 |
  | Increments          |     1 |
  | Multilink           |     0 |
  | Internal            |     3 |
  | Array               |     0 |
  | Foreign             |     0 |
  +---------------------+-------+
  | All                 |    21 |
  +---------------------+-------+

  +-----------------------------------------------------------------------------------------------+
  |     Ticker Events                                                                             |
  +-----------+--------------+--------------+--------------+--------------+--------------+-------+
  | event     |   time calcs |     censored |  occurrences |   per entity | max in queue |    MB |
  +-----------+--------------+--------------+--------------+--------------+--------------+-------+
  | NewPerson |       622150 |            1 |       622149 |  622149.0000 |            1 |     0 |
  | TickEvent |          202 |            0 |          201 |     201.0000 |            1 |     0 |
  +-----------+--------------+--------------+--------------+--------------+--------------+-------+
  | All       |              |              |              |              |              |     0 |
  +-----------+--------------+--------------+--------------+--------------+--------------+-------+
  Note: MB does not include entity members

  +-----------------------------------------------------------------------------------------------+
  |     Ticker Tables                                                                             |
  +--------------+-------+----------+-------+-------+-------+----------+--------------+-------+
  | table        |  rank |    cells | accum | measr | colls | incrmnts |   per entity |    MB |
  +--------------+-------+----------+-------+-------+-------+----------+--------------+-------+
  | TickerEvents |     2 |      603 |     1 |     1 |     0 |      253 |     253.0000 |     0 |
  +--------------+-------+----------+-------+-------+-------+----------+--------------+-------+
  | All          |       |          |       |       |       |          |              |     0 |
  +--------------+-------+----------+-------+-------+-------+----------+--------------+-------+
  Note: MB does not include entity members

  +-----------------------------------------------+
  |     Ticker Members (detail)                   |
  +----------------------------------------+-------+
  | member                                 | bytes |
  +----------------------------------------+-------+
  | Attributes:                            |       |
  |   Built-in:                            |       |
  |     age                                |     8 |
  |     case_seed                          |     8 |
  |     censor_time                        |     8 |
  |     entity_id                          |     4 |
  |     events                             |     2 |
  |     time                               |     8 |
  |   Simple:                              |       |
  |     event_count                        |     4 |
  |     event_type                         |     1 |
  |     exogenous_births                   |     1 |
  |     integer_year                       |     4 |
  |     premier_tic                        |     1 |
  |     prochain_tic                       |     8 |
  |     ps                                 |     1 |
  |     year                               |     1 |
  |   Maintained:                          |       |
  |     endogenous_conceptions             |     1 |
  |   Link:                                |       |
  | Events:                                |       |
  |     NewPerson                          |    24 |
  |     TickEvent                          |    24 |
  | Increments:                            |       |
  |     TickerEvents increment             |    16 |
  | Multilink:                             |       |
  | Internal:                              |       |
  |     TickerEvents (inevent) event_count |     4 |
  |     event_count (lagged)               |     4 |
```

```
    |     event_count (counter at lagged)    |     8 |
    | Array:                                 |       |
    | Foreign:                               |       |
    +----------------------------------------+-------+
    | Sum of member bytes                    |   140 |
    | Bytes per entity                       |   192 |
    | Storage efficiency (%)                 |  72.9 |
    +----------------------------------------+-------+

  ***********************************************
  *  Resource Use Detail for Derived Tables  *
  ***********************************************

    +-------------------------------------------------------------------+
    |      Derived Tables                                               |
    +-------------------------------+-------+----------+-------+-------+
    | derived table                 | rank  |   cells  | measr |    MB |
    +-------------------------------+-------+----------+-------+-------+
    | IM_GMM_KeyInputs              |     1 |        7 |     1 |     0 |
    | IM_IncidenceRR                |     3 |    24480 |     1 |     0 |
    | IM_MajorGeneBirthDistribution |     1 |        6 |     1 |     0 |
    | IM_OncogenesisRR              |     3 |    24480 |     1 |     0 |
    | IM_PolygeneBirthDistribution  |     1 |       51 |     1 |     0 |
    | IM_RiskEvaluationAge          |     0 |        1 |     1 |     0 |
    +-------------------------------+-------+----------+-------+-------+
    | All                           |       |          |       |     0 |
    +-------------------------------+-------+----------+-------+-------+

Resource Use Report - End
```

[back to topic contents]

Model Symbols

Symbols in model code and in the user interface.

This topic is under construction and/or revision.

Related topics

- Model Documentation: Home topic on model documentation
- Symbol Labels and Notes: Human-language labels and notes for model symbols

Topic contents

- Introduction and outline
- Doxygen brief descriptions for model symbols

Introduction and outline

Content to follow.

[back to topic contents]

Doxygen brief descriptions for model symbols

This subtopic describes doxygen brief descriptions created by the OpenM++ compiler for model symbols. An IDE such as Visual Studio can use these doxygen labels to provide information on model symbols used in C++ model code directly from the code editor. Syntactic islands in model code should be hidden so that doxygen and the IDE can correctly parse the C++ portions of the model project, as described in Model Code - Hiding syntactic islands.

This subtopic contains the following sections:

- Background on doxygen
- RiskPaths example
- Examples of doxygen brief descriptions

[back to topic contents]

Background on doxygen

Doxygen is a widely used tool which generates human-readable hyperlinked HTML documentation for a C++ project. Doxygen fully parses the project's C++ source code for symbols and symbol references, and will incorporate descriptive information provided in specially-structured comments in the C++ source code. Here's an example of a structured doxygen comment in the C++ source code of the OpenM++ compiler:

```
class CodeBlock : public list<string>
{
public:
...
   /**
    * Push block of code at the top of the list.
    * No indentation applied (assuming zero indent at the top)
    *
    * @param   push_block The block of code to be inserted.
    */
   void push_header(const CodeBlock & push_block);
...
};
```

In this example the comment block starting with `/**` tells doxygen to parse the comment block for structured descriptive text about the C++ symbol whose declaration or definition follows in the code. Doxygen takes the first line of the comment block as a brief description of the symbol `push_header`.

Doxygen recognizes several ways to supply information in structured comments. For example, the following supplies only the doxygen 'brief description' for `push_header`:

```
class CodeBlock : public list<string>
{
public:
...
   /// Push block of code at the top of the list.
   void push_header(const CodeBlock & push_block);
...
};
```

Doxygen is so widely used that some IDEs (e.g. Visual Studio) scan a project's C++ source code for doxygen comments to improve functionality. For example, in the Visual Studio C++ project for the OpenM++ compiler, hovering the cursor over the symbol `push_header` in line 347 of the module `CodeGen.cpp` causes the IDE to display a pop-up which includes information extracted from the doxygen comment for `push_header` which the IDE found elsewhere in the project:

[back to doxygen brief descriptions for model symbols]
[back to topic contents]

RiskPaths example

The OpenM++ compiler generates C++ code which declare C++ symbols to implement model symbols declared in syntactic islands in model code.

For example, the model code which declares the `unions` attribute of the `Person` entity in the `Unions.mpp` module in `RiskPaths` is

```
actor Person
{
   //EN Union counter
   int unions = {0};
...
```

The OpenM++ compiler parses this code and creates the following C++ code in `om_declarations.h` to implement the `unions` attribute:

```
class Person : public Entity<Person>
{
public:
...
   /// attribute(simple) int: Union counter
   unions_om_type unions;
...
```

The OpenM++ compiler generated a line of C++ code to declare `unions` as well as a preceding doxygen comment containing the doxygen brief description. The doxygen brief description = generated by the OpenM++ compiler has two parts. The first part gives the kind of symbol (a simple attribute) and the the underlying type (`int`). The second part after the `:` is the symbol label provided in model code, in the default language of the model (`EN` for `RiskPaths`).

If the RiskPaths project is opened in Visual Studio and the model built, hovering the cursor over the symbol `unions` in line 148 of the module `Unions.mpp` causes the IDE to display a pop-up for that symbol:

The first line of the popup displays C++ information for `unions`, which can contain useful information like array shape, C++ type, and the containing class. In this example, the first line gives the kind of entity of the `unions` attribute, namely `Person`. The second line of the popup displays the doxygen brief description for `unions` generated by the OpenM++ compiler. In this example, it shows that `unions` is a simple assignable attribute of type int, with label `Union counter`.

[back to doxygen brief descriptions for model symbols]
[back to topic contents]

Examples of doxygen brief descriptions

The following table shows, for each kind of model symbol, an example from `RiskPaths` and the doxygen brief description generated by the OpenM++ compiler. The brief description will appear in a pop-up if the cursor is hovered over the symbol in the Visual Studio IDE, as illustrated above.

Kind of symbol	RiskPaths symbol	Generated brief description	Remarks
attribute	`age`	`attribute(built-in) Time: age`	The type of the built-in attribute `age` is `Time`
attribute	`parity_status`	`attribute(simple) PARITY_STATE: Parity status derived from the state parity`	`parity_status` is a simple assignable attribute of type `PARITY_STATE` (a classification)
attribute	`integer_age`	`attribute(identity) LIFE: Current integer age`	`integer_age` is an identity attribute whose RHS is the expression `COERCE( LIFE, self_scheduling_int(age) )`
classification	`PARITY_STATE`	`Classification {0...1}: Parity status`	The range of possible values is shown.

Kind of symbol	RiskPaths symbol	Generated brief description	Remarks
classification level	`PS_PREGNANT`	`Classification PARITY_STATE(PS_PREGNANT): Pregnant`	The Visual Studio IDE also gives the integer value of the classification level.
range	`LIFE`	`Range {0...100}: Simulated age range`	The range of possible values is shown.
partition	`AGEINT_STATE`	`Partition {0...11}: 2.5 year age intervals`	The range of possible values is shown.
entity function	`Union1DissolutionEvent`	`Implement the event Union1DissolutionEvent when it occurs in the Person entity (model code)`	The label of the function from model code is shown, followed by `(model code)` to indicate the provenance of the function definition.
parameter	`ProbMort`	`Parameter double: Death probabilities`	The Visual Studio IDE also gives the shape of array parameters which for `ProbMort` is 101.

[back to doxygen brief descriptions for model symbols]
[back to topic contents]

Parameter and Table Display and Content

This topic describes how to control the display and presence of parameters and tables using statements in model code.

Related topics

- OpenM++ User Interface
- Model Code

Topic contents

- Parameter groups Organizing parameters into a hierarchy
- Table groups Organizing tables into a hierarchy
- Dual UI Specifying a user interface with switchable simplified and detailed views
- Model trim down Creating a trimmed-down model by removing parameters and tables
- Derived parameters as tables Output derived parameters as tables

Parameter groups

A parameter group is a named, ordered list of parameters and other parameter groups. Parameter groups can be used to organize the parameters of a model into a hierarchical structure for display and navigation in the model UI. A parameter or parameter group can be a component of zero, one, or more than one parameter groups.

In the hierarchical display of input parameters in the model UI, parameters and parameter groups which are not part of any other parameter group are displayed at the root level in lexicographical order by name.

Parameter groups can also be used to identify groups of parameters in other model code statements such as `hide`, `parameters_retain`, or (for derived parameters) `parameters_to_tables`.

The following example declares a parameter group named `PG12_SchoolOneFate` which consists of the model input parameter `Educ1Model` followed by three other parameter groups.

```
parameter_group PG12_SchoolOneFate  //EN Primary School
{
   Educ1Model,
   PG10_SchoolOneFateBase,
   PG11_SchoolOneFateRefined,
   PG10_ShoolOneTracking
};
```

The name of a model code module, quoted, can also be an element of a parameter group. It will be expanded into a list of all the parameters declared in the module, in lexicographical order. For example

```
parameter_group PG0_FertilityParameters  //EN Fertility parameters
{
   "Fertility.mpp"
};
```

declares a parameter group named `PG0_FertilityParameters` containing all parameters declared in the `Fertility.mpp` model code module.

Derived parameters in a `parameter_group` are absent from the hierarchical display of parameters in the model UI. Derived parameters can be displayed in the UI as described below.

Modgen-specific: The Modgen-specific statement `model_generated_parameter_group` is treated as a synonym of `parameter_group` by OpenM++.

[back to topic contents]

Table groups

Table groups are very similar to Parameter groups. They are used to display a model's tables in a hierarchy in the model UI. A table or table group can be a component of zero, one, or more than one table groups.

In the hierarchical display of tables in the model UI, tables and table groups which are not part of any other table group are displayed at the root level in lexicographical order by name.

Table groups are also used to identify groups of tables in other model code statements such as `hide` or `tables_retain`.

Table groups can be used for run-time table selection using model options `Tables.Retain` or `Tables.Suppress`.

The following example declares a table group named `TG04_Education` which consists of three other table groups.

```
table_group TG04_Education //EN Education
{
   TG04_Preschool,
   TG04_Primary,
   TG04_Secondary
};
```

The name of a model code module, quoted, can also be an element of a table group. It will be expanded into a list of all the tables declared in the module, in lexicographical order. For example

```
table_group TG0_AllTables //EN All tables in Tables.mpp
{
   "Tables.mpp"
};
```

declares a table group named `TG0_AllTables` containing all tables declared in the `Tables.mpp` model code module.

[back to topic contents]

Dual UI

The OpenM++ UI can present either a simplified or a detailed model interface to the user, and the user can switch between the two dynamically in the UI by tapping a button. The simplified interface can contain fewer parameters and tables than the detailed interface. Which parameters and tables are displayed in each interface is specified in model source code using one or more `hide` or `show` statements. A model can contain either `hide` statements or `show` statements, but not both. If a model contains no `hide` or `show` statements, it has a single interface and the button to choose the simplified or detailed interface is absent from the UI. If a model has both interfaces, the simplified interface is displayed by default.

The `hide` statement syntax is like:

```
hide P02_Fertility, TG01_Life_Tables;
```

The arguments to `hide` can be the names of tables, parameters, or groups.

The `show` statement has the same syntax. The `show` statement hides all parameters, tables, and groups except those listed as arguments to `show` statements.

`hide` and `show` do not change which parameters or tables are present in the model. They should not be confused with suppress or retain statements in model code which burn in parameters or remove tables from the model itself when it is built: `parameters_retain`, `parameters_suppress`, `tables_retain`, `tables_suppress`.

`hide` and `show` should also not be confused with the run-time model options `Tables.Suppress` and `Tables.Retain` which specify which tables are output in a model run.

Modgen-specific: The Modgen `hide` syntax which surrounds arguments in parentheses is also recognized, and treated as described in the description of `hide` above. Modgen `hide` functionality is similar but not equivalent to ompp `hide` functionality. Modgen `hide` of a table suppresses it from the model, and is similar to the ompp `tables_suppress` statement. Modgen `hide` of a parameter does not remove it, but instructs the UI to not display it.

[back to topic contents]

Model trim down

A family of four model code statements can be used to trim down a model at build time by selectively suppressing parameters using `parameters_suppress` or tables using `tables_suppress`. Suppression of parameters or tables does not affect the simulation. The complementary statements `parameters_retain` and `tables_retain` specify that the model is only to contain specified parameters or tables, suppressing all others. Suppress and retain are mutually exclusive: The OpenM++ compiler will raise an error if model code contains both `parameters_suppress` and `parameters_retain` statements, or both `tables_suppress` and `tables_retain` statements.

Suppressed parameters are burned into the executable using values published when the model is built. Suppressed parameters are absent from the user interface and the model database, and from metadata in the database. Large models can benefit both in build time and start-up time by suppressing parameters, because there is no need to read suppressed parameters from the database when launching the model. Suppressing parameters can also simplify the UI of a deployed model.

Suppressed tables are completely removed from the model. Large models can benefit both in build time and run time by suppressing tables.

Table dependencies specified using the `dependency` statement are nevertheless respected if a suppressed table is required by a non-suppressed table. Suppressed tables which are required by other tables are computed internally but are otherwise invisible.

Branches of the parameter or table hierarchy which become empty because of parameter or table suppression are suppressed from the model metadata and the user interface.

The following example is an extract from a model code module `SuppressRetain.ompp` which was added to the large OncoSim model to create a trimmed-down test version of the model which contained only parameters and tables related to breast cancer.

```
parameters_retain
   SimulationSeed,
   SimulationCases,
   Breast_Cancer_Parameters
;

tables_retain
   TG01_Breast_Cancer_Tables
;
```

OpenM++ also includes the ability to selectively suppress tables at run-time using the model run options `Tables.Suppress` and `Tables.Retain`. These options allow a model user to economize processing time and storage by restricting output to specific tables of interest from the available tables in the model.

Unlike the model run options `Tables.Suppress` and `Tables.Retain`, the model code statements `tables_suppress` and `tables_retain` remove tables completely from a model. That can improve model build time, run time, and run storage, but tables suppressed at build time are not available to users at run time.

A model with suppressed parameters builds faster because its metadata and Default values are not published to the model database. The model also launches faster because there is no need for it to read the suppressed parameters from the database when the model starts. A suppressed parameter can be made visible and editable in the model UI by changing the suppress/retain statement and rebuilding the model. This can be simpler than using the Fixed parameter mechanism which requires moving the file containing the parameter values between two folders.

Models can contain diagnostic tables used for testing and development, but which are only needed occasionally subsequently. Instead of commenting out or removing such tables, they can be kept, but added to a table group and then suppressed using `tables_suppress`. Doing so ensures that the diagnostic tables continue to be parsed and verified when the model is built, without imposing additional complexity or costs to the published model.

During model development, a model is often modified, built, and run repeatedly when working on a specific component. That iterative development process can be accelerated by using `parameters_retain` and `tables_retain` temporarily to focus only on the parameters and tables associated with the current development activity. That optimizes the model to the current development activity without changing the simulation logic. After the development activity is complete, the temporary `parameters_retain` and `tables_retain` statements can be removed.

[back to topic contents]

Derived parameters as tables

Summary

A derived parameter is normally invisible in model inputs and outputs but can be made visible by exporting it as a derived table using the `parameters_to_tables` statement.

The argument of `parameters_to_tables` is a comma separated list of derived parameters or groups of derived parameters. Model code can contain multiple occurrences of `parameters_to_tables`.

The corresponding derived table

- has the same name as the derived parameter
- has the same metadata as the derived parameter including parameter label, note, dimension labels, and dimension names

- converts parameter values to `double`, with classifications, ranges, and partitions converted to `{0,1,2,...}` and `bool` converted to `{0,1}`, where `0` is `false` and `1` is `true`.
- has an implicit dimension for sub/member/replicate for runs with multiple subs
- computes, for overall run results, the average across subs (like other derived tables)

A derived table created by `parameters_to_tables` acts like other derived tables and can be

- displayed in the UI as a multi-dimensional table
- organized in the hierarchical display of tables in the model UI using `table_group`
- suppressed or retained in model outputs using `tables_suppress` or `tables_retain`
- exported in `csv` format for downstream analysis either from the UI or by using `dbcopy`
- used in model output comparisons with `test_models`

Exposition and example

A derived parameter (aka model-generated parameter) is declared using the `derived` keyword. It is computed by model code before the simulation starts using values of other parameters.

Modgen-specific: In Modgen, derived parameters are declared using the keyword `model_generated`. OpenM++ treats `model_generated` as a synonym of `derived`. In Modgen, derived tables are declared using the keyword `user_table`. OpenM++ treats `user_table` as a synonym of `derived_table`.

For example, here's the declaration of the derived parameter `ImmigrantDonors` in the `OzProj` model:

```
parameters
{
  ...
  //EN Number of immigrant donors in initial population
  model_generated int ImmigrantDonors;
  ...
};
```

`OzProj` computes the value of `ImmigrantDonors` in the function `PersonCore_PreSimulation` by counting microdata input records which satisfy particular conditions. The computation depends on the input parameter `MicroDataInputFile` which gives the name of the file containing input microdata.

Derived parameters are not editable and are not present in the hierarchical display of parameters in the model UI. However, `parameters_to_tables` makes selected derived parameters visible in the UI by converting them to derived tables.

The following statement in `OzProj` makes the 7 derived parameters in `OzProj` visible as identically-named derived tables in the model UI.

```
parameters_to_tables
    EmigrationHazard,
    FertilityHazard,
    MortalityHazard,
    ImmigrantDonors,
    EmigrationHazard,
    FertilityHazard,
    MortalityHazard
;
```

Every parameter specifies the type of its value(s), e.g. `double`, `int`, `bool`, `REGION`, `AGE_GROUP`. Because the value of a table cell is always `double`, `parameters_to_tables` may need to convert the parameter value type to `double` in the derived table. The conversion follows normal C++ type conversion rules. This includes converting parameter values of type `Range`, `Classification` or `Partition` to `{0,1,2,...}`, and parameter values of type `bool` to `{0,1}`, where `0` is `false` and `1` is `true`.

Derived parameters transformed to derived tables can be members of a table group, e.g.

```
table_group DerivedParameters //EN Derived Parameters
{
    EmigrationHazard,
    FertilityHazard,
    MortalityHazard,
    ImmigrantDonors,
    EmigrationHazard,
    FertilityHazard,
    MortalityHazard
};
```

This allows them to be organized hierarchically in the UI, or suppressed/retained as a group using `tables_suppress` or `tables_retain`.

[back to topic contents]

Population Size and Scaling

This topic describes ways a model can specify the size of the simulation population, the size of the real-world population represented by the simulation population, and the scaling of table results to the size of the real-world population.

Related topics

- Use Modules: The `use` statement and supplied library of `use` framework modules

Topic contents

- Introduction
- Population size and scaling - case-based model
- Population size and scaling - time-based model

Introduction

The following concepts are used in this topic.

Term	Explanation
run	An execution of a model. A run can consist of one or more independent subs/members/replicates.
member/replicate/sub/subsample	Synonyms denoting each of the independent simulations which together constitute a run.
case	An independent, isolated entity or small collection of related entities such as a family or clan.
case-based model	A model which simulates, in each sub/replicate, a population of independent, non-interacting cases.
time-based model	A model which simulates, in each sub/replicate, a population of interacting entities.
simulation population	The population of entities in a model simulation.
real population	The real-world population represented by a simulation population.
`use` module	A code module supplied with OpenM++ which implements underlying functionality for a model.

An OpenM++ model specifies population size and scaling in different ways depending on what kind of model it is. For example, a model which creates a synthetic population from multivariate distributions has no intrinsic size whereas a model based on a micro-data file may have its size fixed to the number of observations in that file.

To support different model frameworks OpenM++ includes a library of source code modules. Selected modules from this library are incorporated into a model through `use` statements to implement the desired model framework. These `use` statements are usually found in a model code module named `ompp_framework.ompp`. For more information, please see the Use Modules topic.

[back to topic contents]

Population size and scaling - case-based

Population size and scaling is specified for a case-based model by selecting a pair of modules from the `use` library, one which determines population size and one which determines population scaling. For example, the following code fragment from the `ompp_framework.ompp` module for a model specifies that population size is given for the entire run (rather than for each replicate of the run), and that the results are not scaled to a real population.

```
use "case_based/case_based_cases_per_run_exogenous.ompp";
use "case_based/case_based_scaling_none.ompp";
```

The following table lists the available choices to specify population size for a case-based model:

`use` module	Description

`use` module	Description
`case_based/case_based_cases_per_member_exogenous.ompp`	The number of cases in a member is given by the input parameter `CasesPerMember` which is declared in this module. The number of cases in the run is the product of `CasesPerMember` and the number of replicates specified for the run.
`case_based/case_based_cases_per_run_exogenous.ompp`	The number of cases for the entire run is given by the input parameter `SimulationCases` which is declared in this module. The cases are evenly divided among the members specified for the run. Any 'extra' cases are assigned to lower-numbered members.
`case_based/case_based_cases_per_member_endogenous.ompp`	The number of cases in a member is given endogenously by the derived parameter `CasesPerMember`, which is declared in this module. `CasesPerMember` is set by a `PreSimulation` function in model code. `CasesPerMember` might, for example, be set to the number of observations in a micro-data input file. The number of cases in the run is the product of `CasesPerMember` and the number of replicates specified for the run. *NOT YET IMPLEMENTED*
`case_based/case_based_cases_per_run_endogenous.ompp`	The number of cases for the entire run is given endogenously by the derived parameter `SimulationCases`, which is declared in this module. `SimulationCases` is set by a `PreSimulation` function in model code. `SimulationCases` might, for example, be set to the number of observations in a micro-data input file. Model code would allocate observations in the micro-data file evenly to each of the members of the run, and allocate any 'extra' cases to lower-numbered members. *NOT YET IMPLEMENTED*

The following table shows the available choices to specify population scaling for a case-based model:

`use` module	Description
`case_based/case_based_scaling_exogenous.ompp`	The simulated population in each sub/member/replicate is scaled to represent the real-world population given by the input parameter `SimulatedPopulation`, which is declared in this module. Incorporates weights if model uses weighted tabulation.
`case_based/case_based_scaling_endogenous.ompp`	The simulated population in each sub/member/replicate is scaled to represent a real-world population provided by model code by a call to the global function SetPopulation(). Incorporates weights if model uses weighted tabulation.
`case_based/case_based_scaling_none.ompp`	The simulated population in each sub/member/replicate is scaled to the number of cases in the run. For a run with a single member, this is equivalent to no population scaling. For a run with more than one member, each member is scaled up by the number of members in the run.
`case_based/case_based_scaling_endogenous_or_none.ompp`	Acts like `case_based_scaling_endogenous` if the parameter `DisablePopulationScaling` is `false`, or like `case_based_scaling_none` if `true`.
`case_based/case_based_scaling_disable.ompp`	Disables population scaling. Each sub is scaled up by the number of subs, which causes overall (average) run results to become the sums of the original (unscaled) subs.

[back to topic contents]

Population size and scaling - time-based

Population size and scaling is specified for a time-based model by selecting a module from the `use` library which determines both population size and scaling. All such modules declare a parameter `StartingPopulationSize` which gices the size of the starting population. The size of the starting population is the same for all members of a time-based model run.

The following code fragment from the `ompp_framework.ompp` module for a model specifies that results are not scaled.

```
use "time_based/time_based_scaling_none.ompp";
```

The following table shows the available choices to specify population size scaling for a time-based model:

`use` module	Description
`time_based/time_based_scaling_exogenous.ompp`	The simulated population in each member/replicate is scaled to represent the real-world population given by the input parameter `SimulatedPopulation`, which is declared in this module.
`time_based/time_based_scaling_endogenous.ompp`	The simulated population in each member/replicate is scaled to represent the real-world population given by the input parameter `SimulatedPopulation`, which is declared in this module. Or maybe by a call to the function SetPopulation(). *NOT FULLY TESTED*
`time_based/time_based_scaling_none.ompp`	The simulated population in each member/replicate is not scaled.

[back to topic contents]

Screened Tables

Screened tables let model developers implement Statistical Disclosure Control (SDC) policies at the cell level of entity tables. Screened tables can also be used for output quality management, for example to suppress statistically unreliable cells or to round values to a given number of digits of precision to avoid the impression of spurious accuracy in model outputs.

Related topics

- Entity Tables
- Model Code

Topic contents

- Introduction Introduction
- Overview Overview
- Syntax and simple example Syntax and simple example
- Remarks and limitations Remarks and limitations
- Extrema Collections Extrema collections
- Screening function arguments Arguments of the screening transformation function(s)
- Examples Examples
- Annex 1: SM1 Code
- Annex 2: SM1 Parameters

Introduction

Please see the following references for an overview of Statistical Disclosure Control:

- Statistical disclosure control (Wikipedia)↗
- The SACRO guide to statistical output checking↗

[back to topic contents]

Overview

Screened Tables is an optional capability that allows a model developer to examine and modify table cell values, based on the values themselves and their provenance. It is activated by using a keyword in the entity table declaration, and by providing a C++ function which implements screening at the cell level. That C++ screening function is not necessarily complex, and multiple function arguments are supplied to ease the task. This wiki topic also includes a suite of working examples.

The examples are based on the model `SM1` in the OpenM++ distribution at `OM_ROOT/models/SM1`. `SM1` is a simple model which adds several attributes to the `NewCaseBased` model to provide raw material for the examples in this topic.

The example table declarations in this topic can be copy/pasted directly into the module `SM1/code/ScreenedTables.ompp`, and the corresponding example screening code can be copy/pasted directly into the body of the function `TransformScreened1` in `SM1/code/ScreeningCode.ompp`. The upper right corner of each code block in this wiki topic should display a browser pop-up which can be clicked to copy the entire block to the system clipboard for subsequent pasting into `SM1`, for in-depth exploration.

To provide flexibility and facilitate comparison of different methods a model can use up to four different screening functions. The examples in this topic use only a single method, method #1.

[back to topic contents]

Syntax and simple example

To activate one of the four screening methods for a given entity table, include exactly one of the four keywords `screened1`, `screened2`, `screened3`, or `screened4` in the table's properties, and supply a definition for the corresponding C++ transformation function. For example, the following table is

declared in `SM1/code/ScreenedTables.ompp`:

```
table snapshot screened1 Person ExampleTable
[trigger_entrances(integer_age, 50)]
{
  {
    unit,          //EN Persons
    mean(earnings), //EN Average earnings
    P50(earnings)   //EN Median earnings
  }
};
```

The table property `screened1` specifies that it will be screened using screening method #1, which uses the C++ transformation function `TransformScreened1`.

That means that each of the three accumulators (statistics) in the single cell of `EarningsAt50` will be subject to modification by the developer-supplied C++ function `TransformScreened1`. This function is called automatically when the simulation of each sub/member/replicate completes, for each value in this table.

The developer-supplied function `TransformScreened1` takes 10 arguments (described below), whose values are supplied by the framework. The first argument `in_value` is a value in the table before modification. The function returns the possibly modified value. Here's an example function body of `TransformScreened1` defined in the model code module `SM1/code/ScreeningCode.ompp`:

```
double TransformScreened1(
  const double in_value,
...
)
{
  /// transformed value, initialized to quiet NaN (shows as empty)
  double out_value = UNDEF_VALUE;

  // notional example of transformation (round to 100's)
  out_value = 100.0 * std::round(in_value / 100.0);

  return out_value;
}
```

The C++ code in this example rounds all table values to 100's.

Here's a comparison of the table values for both unscreened and screened versions of the table:

Quantity	Label	unscreened	screened
`unit`	Persons	2538	2500
`mean(earnings)`	Average earnings	99203.9	99200
`P50(earnings)`	Median earnings	101191	101200

To examine the transformation process in action, build the Debug version of `SM1` and set breakpoints in the function `TransformScreened1` in the module `ScreeningCode.ompp`.

[back to topic contents]

Remarks and limitations

- only at sub level, indirectly at run level
- only at cell level (including marginal cells if present) - no checks for residual disclosure
- only at accumulator/statistic level, indirectly at expression/measure level.
- only for entity tables, indirectly to derived tables.
- use the statistic `mean` to screen averages, i.e. `mean(x)` instead of `sum(x)/unit` as in older model code.
- to easily deactivate screening without editing any table declarations, insert the single line `return in_value;` as the first line in the screening function.
- in use cases where model code is released but the screening algorithm is confidential, the screening function can be built separately and

supplied as an object file or library to build the model. Alternatively, an alternative module containing a 'do nothing' screening function can be distributed, if the model code is distributed with a non-confidential synthetic version of the microdata file(s).

[back to topic contents]

Extrema collections

The screening function arguments `smallest` and `largest` are collections containing the highest M and lowest M observations in the table cell containing the value being screened, where M is a configurable constant for each of the four screening methods. These collections allow implementing 'dominance' rules for cell suppression, e.g. suppress a cell total if the top 3 observations in the cell account for more than 70% of the total. To set an appropriate value for M, use the corresponding option `screened[1-4]_extremas_size` in model code. For example, the following statement retains the highest 3 and lowest 3 observations in the `smallest` and `largest` extrema collections for method #1:

```
options screened1_extremas_size = 3;
```

Extrema collections might be smaller than the specified size if there are fewer than that many observations in the cell.

Extrema collections can contain the special floating point values +inf and -inf. They never contain the special floating point value NaN, because a NaN increment is treated as a run-time model error by OpenM++.

Only certain statistics are considered eligible for extrema collections. They are

`sum`
`minimum`
`maximum`
`mean`

If a quantity is ineligible the associated extrema collections will be empty.

Reducing the size of extrema collections reduces memory and processing requirements.

M is set to 0 by default for all four screening methods, to avoid the computational and memory overhead of maintaining these collections for each cell of each screened table unless needed by the screening method.

[back to topic contents]

Screening function arguments

This subtopic contains several reference sections which are listed and linked here for convenience:

- Screening function signature
- statistic enumeration
- increment enumeration
- table enumeration
- attribute enumeration

This subtopic describes the arguments to the developer-supplied screening function(s).

It is essential that the definition of the screening function(s) in model code have the correct argument types. Otherwise the model build will fail at the C++ link stage, with an error message like

```
error LNK2019: unresolved external symbol "double __cdecl TransformScreened1(...
```

The correct function signature can be copied from the section below, or from the file `OM_ROOT/include/omc/globals1.h` in the OpenM++ distribution.

The rows of the following table describe the 10 arguments of a screening function. The example column contains values pasted from a debugger session on a breakpoint in the function `TransformScreened1` in `SM1`, on the second invocation of the function, using the table given earlier.

Name	Example	Notes

Name	Example	Notes
`in_value`	`99203.855397951411`	The original value in the table, which can be transformed or suppressed by code in the transformation function.
`description`	`ExampleTable: accumulator 1: mean(value_out(interval(earnings)))`	A descriptive string which can be useful in debugging sessions. Don't attempt to parse it for content. Instead, use the enumerator arguments described in rows below.
`statistic`	`mean (4)`	The enumerator for the statistic for use in function code, e.g. `omr::stat::mean`
`increment`	`value_out (7)`	The enumerator for the increment for use in function code, e.g. `omr::incr::value_out`. For the `unit` keyword (count of increments), the value is `omr::incr::unused`.
`table`	`ExampleTable (0)`	The entity table name as an enumerator, e.g. `omr::etbl::ExampleTable`
`attribute`	`earnings (4)`	The attribute name as an enumerator, e.g. `omr::attr::earnings`. If the attribute name is not visible, e.g. `duration()`, the enumerator is `omr::attr::om_none`.
`observations`	`2538.0000000000000`	The number of observations (increments) in the cell. It is always unweighted, even if the table is weighted. The value is less than the 5,000 cases in the `Default` `SM1` run due to mortality before age 50.
`extrema_size`	`3`	Identical to the value supplied in the `screened[1-4]_extremas_size` option, for use in function code. This is the maximum possible size of the extrema collections. The actual size may be less if there are fewer observations in the cell.
`smallest`	`{0.0000000000000000, 0.0000000000000000, 0.0000000000000000}`	The three smallest observed earnings, in increasing order. The extrema collection is one of the standard C++ container types, specifically `std::multiset<double>`. See code examples elsewhere in this topic for use. The values are all zero in this example because the distribution of earnings in `SM1` is mixed discrete-continuous, with a large subpopulation having zero earnings.
`largest`	`{370810.00000000000, 398272.00000000000, 484007.00000000000}`	The three largest observed earnings, in increasing order.

[back to topic contents]

Screening function signature

```
/**
 * Table screening transformation function #1
 *
 * @param   in_value     The table value subject to transformation.
 * @param   description  A formatted string describing the table and statistic.
 * @param   statistic    The statistic of the accumulator, e.g. sum, mean.
 * @param   increment    The increment of the accumulator, e.g. delta, value_out.
 * @param   table        The table of the accumulator (model-specific).
 * @param   attribute    The attribute of the accumulator (model-specific).
 * @param   observations The count of observations in the cell (# of increments).
 * @param   extrema_size The maximum size M of the two extrema collections (configurable)
 * @param   smallest     The extrema collection containing the smallest M observations.
 * @param   largest      The extrema collection containing the largest M observations.
 *
 * @returns The transformed version of in_value.
 */
double TransformScreened1(
    const double in_value,
    const char* description,
    const omr::stat statistic,
    const omr::incr increment,
    const omr::etbl table,
    const omr::attr attribute,
    const double observations,
    const size_t extrema_size,
    const std::multiset<double>& smallest,
    const std::multiset<double>& largest
)
```

[back to screening function arguments]
[back to topic contents]

statistic enumeration

This enumeration is generated by the OpenM++ compiler in the file `src/om_types0.h`.

```
namespace omr {
  /// statistic in an entity table
  enum class stat {
    unit,
    sum,
    minimum,
    maximum,
    mean,
    variance,
    stdev,
    P1,
    P2,
    P5,
    P10,
    P20,
    P25,
    P30,
    P40,
    P50,
    P60,
    P70,
    P75,
    P80,
    P90,
    P95,
    P98,
    P99,
    gini,
  };
} // namespace omr
```

[back to screening function arguments]
[back to topic contents]

increment enumeration

This enumeration is generated by the OpenM++ compiler in the file `src/om_types0.h`.

```
namespace omr {
  /// increment in an entity table
  enum class incr {
    unused,
    delta,
    delta2,
    nz_delta,
    value_in,
    value_in2,
    nz_value_in,
    value_out,
    value_out2,
    nz_value_out,
  };
} // namespace omr
```

[back to screening function arguments]
[back to topic contents]

table enumeration

This model-specific enumeration is generated by the OpenM++ compiler in the file `src/om_types0.h`. Here it is for the `SM1` model:

```
namespace omr {
  /// entity table in model
  enum class etbl {
    ExampleTable,
    om_none
  };
} // namespace omr
```

[back to screening function arguments]
[back to topic contents]

attribute enumeration

This model-specific enumeration is generated by the OpenM++ compiler in the file `src/om_types0.h`. Here it is for the `SM1` model:

```
namespace omr {
  /// visible entity attribute in model
  enum class attr {
    age,
    alive,
    all_earnings,
    benefit,
    case_id,
    case_seed,
    earnings,
    entity_id,
    integer_age,
    lifecycle_counter,
    lifecycle_event,
    positive_earnings,
    region,
    se_earnings,
    time,
    under_audit,
    om_none
  };
} // namespace omr
```

[back to screening function arguments]
[back to topic contents]

Examples

This subtopic contains some worked examples of screened tables and screening functions.

Each example shows the table declaration, the body of the screening function, and a cell-by-cell comparison of the effects of screening on the table.

The examples are meant to illustrate coding approaches to different kinds of screening requirements. They have not been tested carefully for validity.

Some of these examples use standard math functions which require the header file `<cmath>`, which is made available to model code through a `#include` instruction in the file `SM1/code/custom_early.h`.

- Example 1 Rounding based on the kind of statistic
- Example 2 Rounding to 3 digits of precision
- Example 3 Suppressing cells with few observations
- Example 4 Suppressing cells dominated by a few large observations

[back to topic contents]

Example 1

This example rounds values to a fixed number of decimal digits, but treats different statistics differently. Specifically, the gini coefficient is not modified, counts are rounded to 100's, means are rounded to 1000's, sums are rounded to 1000000's, and other statistics are suppressed.

Table declaration:

```
table snapshot screened1 Person ExampleTable
[trigger_entrances(integer_age, 50)]
{
  {
    unit,                  //EN Persons
    nz_value_out(earnings), //EN Persons with earnings
    mean(earnings),        //EN Average earnings
    sum(earnings),         //EN Total earnings
    P50(earnings),         //EN Median earnings
    maximum(earnings),     //EN Maximum earnings
    gini(earnings)         //EN gini of earnings
  }
};
```

Screening function body:

```
{
   /// transformed value, initialized to quiet NaN (shows as empty)
   double out_value = UNDEF_VALUE;

   /// the increment is from the count-like nz family
   bool is_nz =
         (increment == omr::incr::nz_delta)
      || (increment == omr::incr::nz_value_in)
      || (increment == omr::incr::nz_value_out);

   if (statistic == omr::stat::gini) {
      // gini coefficient
      // do not modify
      out_value = in_value;
   }
   else if ((statistic == omr::stat::unit) || is_nz) {
      // count-like value
      // round to 100's
      out_value = 100.0 * std::round(in_value / 100.0);
   }
   else if (statistic == omr::stat::mean) {
      // average-like value
      // round to 1000's
      out_value = 1000.0 * std::round(in_value / 1000.0);
   }
   else if (statistic == omr::stat::sum) {
      // total-like value
      // round to 1000000's
      out_value = 1000000.0 * std::round(in_value / 1000000.0);
   }
   else {
      // suppress other things
      out_value = UNDEF_VALUE;
   }

   return out_value;
}
```

Effects: Quantity | Label | unscreened | screened ----------|----------|----------:|---------: `unit` | Persons | 2538 | 2500 `nz_value_out(earnings)` | Persons with earnings | 2030 | 2000 `mean(earnings)` | Average earnings | 99203.9 | 99000 `sum(earnings)` | Total earnings | 251779000 | 252000000 `P50(earnings)` | Median earnings | 101191 | `maximum(earnings)` | Maximum earnings | 484007 | `gini(earnings)` | gini of earnings | 0.375879 | 0.375879

[back to Examples]
[back to topic contents]

Example 2

This example rounds values to a fixed number of digits of precision (3). The table declaration is identical to example 1 immediately above.

Table declaration:

```
table snapshot screened1 Person StatsAt50
[trigger_entrances(integer_age, 50)]
{
   {
      unit,                  //EN Persons
      nz_value_out(earnings), //EN Persons with earnings
      mean(earnings),         //EN Average earnings
      sum(earnings),          //EN Total earnings
      P50(earnings),          //EN Median earnings
      maximum(earnings),      //EN Maximum earnings
      gini(earnings)          //EN gini of earnings
   }
};
```

Screening function body:

```
{
    // pass through non-finite values and 0.0
    if (!std::isfinite(in_value) || in_value == 0.0) {
        return in_value;
    }

    /// transformed value, initialized to quiet NaN (shows as empty)
    double out_value = UNDEF_VALUE;

    // number of significant digits to retain
    const static double precision = 3;

    double d = std::ceil(std::log10(std::abs(in_value)));
    /// power of 10 for scaling
    double power = precision - std::trunc(d);
    /// scaling needed before rounding
    double magnitude = std::pow(10.0, power);

    out_value = std::round(in_value * magnitude) / magnitude;

    return out_value;
}
```

Effects:

Quantity	Label	unscreened	screened
unit	Persons	2538	2540
nz_value_out(earnings)	Persons with earnings	2030	2030
mean(earnings)	Average earnings	99203.9	99200
sum(earnings)	Total earnings	251779000	252000000
P50(earnings)	Median earnings	101191	101000
maximum(earnings)	Maximum earnings	484007	484000
gini(earnings)	gini of earnings	0.375879	0.376

[back to Examples]
[back to topic contents]

Example 3

This example rounds counts to 5's, suppresses cells with under 100 observations. The table has a classification dimension and a margin.

Table declaration:

```
table snapshot screened1 Person ExampleTable //EN High earners by region
[trigger_entrances(integer_age, 50)]
{
    region+
    * {
        high_earner //EN High earners
    }
};
```

Screening function body:

```
{
   /// transformed value, initialized to quiet NaN (shows as empty)
   double out_value = UNDEF_VALUE;

   if (observations < 100) {
      // suppress if fewer than 100 observations
      out_value = UNDEF_VALUE;
   }
   else {
      // round to 5's
      out_value = 5.0 * std::round(in_value / 5.0);
   }

   return out_value;
}
```

Effects: Region | observations | unscreened | screened --------|----------:|----------:|---------: `0` | 1374 | 269 | 270 `1` | 675 | 141 | 140 `2` | 373 | 83 | 85 `3` | 59 | 5 | `4` | 57 | 9 | `All` | 2538 | 507 | 505

[back to Examples]
[back to topic contents]

Example 4

This example implements a dominance rule to suppress cells which are dominated by only a few observations in a cell. Specifically, average earnings are suppressed if the top 3 earners account for 60% or more of the earnings in the cell. The code for the screening function also illustrates how to specialize for a specific table.

Table declaration:

```
table snapshot screened1 Person ExampleTable //EN Average earnings of high earners
[trigger_entrances(integer_age, 50) && high_earner]
{
   region+
   * {
      mean(earnings) //EN Earnings
   }
};
```

Screening function body:

```
{
   /// transformed value, initialized to quiet NaN (shows as empty)
   double out_value = UNDEF_VALUE;

   switch (table) {
   case omr::etbl::ExampleTable:
   {
      assert(extrema_size == 3); // code below requires that extrema size is 3

      assert(statistic == omr::stat::mean);     // sanity check
      assert(attribute == omr::attr::earnings); // sanity check

      double sum_top3 = 0.0;
      for (auto& val : largest) {
         sum_top3 += val;
      }
      double sum_all = in_value * observations;

      if ((sum_top3 / sum_all) >= 0.60) {
         // suppress if top 3 earners account for 60% or more of earnings in cell
         out_value = UNDEF_VALUE;
      }
      else {
         // round value to 1000's
         out_value = 1000.0 * std::round(in_value / 1000.0);
      }
      break;
   }
   default:
   {
      // code to handle other screened tables would go here
      break;
   }
   } // switch
   return out_value;
}
```

Effects: Region | observations | unscreened | screened --------|----------:|----------:|---------: `0` | 236 | 172678 | 173000 `1` | 127 | 168087 | 168000 `2` | 74 | 168747 | 169000 `3` | 5 | 173615 | `4` | 9 | 210466 | 210000 `All` | 451 | 171504 | 172000

Note that the number of observations is small because the table is filtered on high earners.

[back to Examples]
[back to topic contents]

Annex 1: SM1 Code

Contents of the module `SM1/code/Income.ompp`:

```
/* NOTE(Income.mpp, EN)
   This module contains hard-coded notional income dynamics for testing screened tables.
*/

#include "omc/optional_IDE_helper.h" // help an IDE editor recognize model symbols

#if 0 // Hide non-C++ syntactic island from IDE

range REGION //EN Region
{
   0,
   4
};

parameters
{
   double EarningsNonZeroProportion;
   double EarningsScaleFactor;
   double EarningsSigma;
   double SE_EarningsNonZeroProportion;
   double SE_EarningsScaleFactor;
   double SE_EarningsSigma;
   double HighIncomeThreshold;
   double GuaranteedAnnualIncome;
   double AuditThreshold;
   cumrate RegionDistribution[REGION];
};

entity Person
{
   //EN Earnings
   double earnings = { 0.0 };

   //EN Self-employed earnings
   double se_earnings = { 0.0 };

   //EN All earnings
   double all_earnings = earnings + se_earnings;

   //EN Positive earnings
   double positive_earnings = max(0.0, all_earnings);

   //EN Benefit
   double benefit = max(GuaranteedAnnualIncome, GuaranteedAnnualIncome - positive_earnings);

   //EN High earner
   bool high_earner = (all_earnings >= HighIncomeThreshold);

   //EN Under audit
   bool under_audit = (all_earnings >= AuditThreshold) || (positive_earnings != all_earnings);

   //EN Region
   REGION region;

   //EN Notional model of earnings
   void AssignEarnings(void);

   //EN Assign region
   void AssignRegion(void);

   // call EarningsGrowth at each change in integer_age
   hook AssignEarnings, trigger_changes(integer_age);

   // call AssignRegion at Start
   hook AssignRegion, Start, 1;
};

#endif // Hide non-C++ syntactic island from IDE

void Person::AssignEarnings(void)
{
   if (integer_age == 20) {
```

```
    if (integer_age == 20) {
        // Assign starting earnings at age 20
        if (RandUniform(10) < EarningsNonZeroProportion) {
            double z = RandNormal(11);
            double x = EarningsScaleFactor * std::exp(EarningsSigma * z);
            earnings = std::round(x);
        }
        // else earnings have default value of 0

        // Assign starting se_earnings at age 20
        if (RandUniform(12) < SE_EarningsNonZeroProportion) {
            // 80% have self-employed earnings
            double z = RandNormal(13);
            double x = SE_EarningsScaleFactor * std::exp(SE_EarningsSigma * z);
            se_earnings = std::round(x);
        }
    }
    else if (integer_age > 20) {
        // Annual change to earnings
        {
            double u = RandUniform(14);
            // rescale uniform to to [0.9, 1.1]
            u = 0.9 + 0.2 * u;
            double x = earnings * u;
            x *= 1.03; // career growth with increasing age
            earnings = std::round(x);
        }
        // Annual change to se_earnings
        {
            double u = RandUniform(15);
            // rescale uniform to [0.9, 1.1]
            u = 0.9 + 0.2 * u;
            double x = se_earnings * u;
            x *= 1.03; // career growth with increasing age
            se_earnings = std::round(x);
        }
    }
    else {
        // No earnings before age 20
    }
}

void Person::AssignRegion(void)
{
    double draw = RandUniform(3);
    int nRegion = 0;
    Lookup_RegionDistribution(draw, &nRegion);
    region = (REGION)nRegion;
}
```

[back to topic contents]

Annex 2: SM1 Parameters

Contents of `SM1/parameters/Default/Income.ompp`:

```
parameters
{
    double EarningsNonZeroProportion     = 0.80;
    double EarningsScaleFactor           = 50000.0;
    double EarningsSigma                 = 0.25;
    double SE_EarningsNonZeroProportion  = 0.80;
    double SE_EarningsScaleFactor        = 40000.0;
    double SE_EarningsSigma              = 0.25;
    double HighIncomeThreshold           = 250000.00;
    double GuaranteedAnnualIncome        = 20000.00;
    double AuditThreshold                = 250000.00;
    cumrate RegionDistribution[REGION] =
    {
        200, 100, 50, 10, 10
    };
};
```

[back to topic contents]

Symbol Labels and Notes

This topic describes human-language labels and notes associated with model symbols. These labels and notes are used in a model's UI, documentation, and the IDE.

Related topics

- Model Documentation: Home topic on model documentation
- Authored Model Documentation: The autonomous authored component of model documentation
- Generated Model Documentation: The symbol reference component of model documentation
- Model Languages: The human languages supported by a model
- Model Symbols: Symbols in model code and in the user interface

Topic contents

- Introduction and outline
- Symbol labels A label provides a one-line short description of a model symbol
- Symbol notes A note provides in-depth information about a model symbol
- Identifying missing symbol documentation Describes options to identify missing symbol documentation

Introduction and outline

This topic describes how to provide human-readable documentation for model symbols using labels and notes. Symbol labels and notes are used in the model's UI and in the model's generated Symbol Reference.

Labels and notes are described in separate subtopics. A final subtopic describes a family of options understood by the OpenM++ compiler which can help identify missing labels, notes, or translations.

[back to topic contents]

Symbol labels

This subtopic is organized into the following sections:

- Symbol labels: introduction
- Symbol labels: inline with declaration, e.g. `//EN`
- Symbol labels: `//LABEL`
- Symbol labels: `//LABEL` with two-part name
- Symbol labels: `//LABEL` with two-part name (explicit short name)
- Symbol labels: `//LABEL` with two-part name (Modgen scheme)
- Symbol labels: `LABEL` text file

[back to topic contents]

Symbol labels: introduction

Labels for model symbols play an important role in the UI for a model, providing comprehensible human-language text for

- parameters,
- parameter dimensions,
- enumerations (classifications, ranges, partitions),

- enumerators of classifications,
- tables,
- table dimensions,
- table expressions,
- parameter groups (parameter hierarchy)
- table groups (table hierarchy),
- attributes.

Labels for model symbols are also used to aid a model developer using C++ symbols in an IDE. The OpenM++ compiler uses a model's labels to construct doxygen comments in generated C++ code, which can be used by an IDE to display contextual pop-ups for symbols in the model's C++ code, as described below.

See the subtopic identifying missing symbol documentation in this topic for functionality to help identify missing labels or translations. This can be helpful when publishing a model for external users.

A symbol label can also be used in the authored component of model documentation if that content uses `GetLabel`, as described here.

[back to symbol labels]
[back to topic contents]

Symbol labels: inline with declaration, e.g. `//EN`

Model symbols are declared in syntactic islands in model code. A model symbol can have a *label* for each human language declared in the `languages` statement. A symbol label can be provided where the symbol is declared using an in-line comment, for example

```
actor Person
{
   //EN Union counter
   int unions = {0};
...
```

To be recognized as a language label, there must be no white space between `//` and the language code, and some white space is required between the language code and the label text. OpenM++ silently ignores unrecognized language codes.

In this example, a label is provided for the symbol `unions` by a comment on the line immediately preceding the symbol declaration. It can also be provided on the same line as the declaration by placing the trailing `//` comment there.

Model code sometimes declares a symbol positionally rather than using a name. A label can be provided for a positionally-delcared symbol on the same line or on the immediately preceding line, just like for symbols with names.
For example the table declaration

```
table Person T01_LifeExpectancy //EN Life Expectancy
{
  {
    unit,          //EN Total simulated cases
    duration(),     //EN Total duration
    duration()/unit //EN Life expectancy decimals=3
  } //EN Quantities
};
```

provides a label for the table itself, for each of the the three expressions in the table, and for the expression dimension of the table, all in the `EN` language of the model.

Notice that the positional location of the table expression dimension is the trailing closing `}` of the expression dimension, after all expressions.

[back to symbol labels]
[back to topic contents]

Symbol labels: `//LABEL`

A symbol label can also be provided by a `//LABEL` comment in model code. This is particularly useful for models which support more than one language.

In the following example, the `EN` label for the `SimulationCases` parameter is provided where it is declared

```
parameters {
   //EN Number of cases in run (over all members)
   long long SimulationCases;
};
```

and the French language version of the label is provided elsewhere using `//LABEL`

```
//LABEL(SimulationCases,FR) Nombre de cas (dans tous les membres de l'execution)
```

To be recognized as a human-language label, there must be no white space between `//` and `LABEL`. Subsequent white space is optional. The arguments to `//LABEL` are the symbol name and the language code, which to be recognized must be one of those declared in the `languages` statement.

A label supplied inline with the symbol declaration in model code takes precedence over a label supplied through a `//LABEL` comment in model code.

OpenM++ silently ignores unrecognized symbol names and language codes in `//LABEL` comments.
Modgen-specific: Modgen treats a `LABEL` comment with an unrecognized symbol or language as an error. Note that OpenM++ has other mechanisms to help model devs identify missing or mistyped labels described here.

[back to symbol labels]
[back to topic contents]

Symbol labels: `//LABEL` with two-part name

The `//LABEL` syntax uses a two-part name to provide a label for the dimension of a parameter or table, for an expression of a table, for the expression dimension of a table, and for classification levels (enumerators).

The first part of a two-part name is the name of the parameter, table, or classification.
The second part of a two-part name denotes the dimension, expression, or enumerator.

The separator for a two-part name in a `LABEL` comment can be either `::` or `.`.
Modgen-specific: Modgen recognizes only `.` as a separator in two-part names.

There are two ways to specify the second part of a two-part name:

- the explicit short name, or
- the Modgen naming scheme.

[back to symbol labels]
[back to topic contents]

Symbol labels: `//LABEL` with two-part name (explicit short name)

In the following example, the dimensions and expressions of the table `TotalPopulationByYear` have been given explicit short names:

```
table Person TotalPopulationByYear //EN Life table
{
  //EN Curtate age
  age => integer_age +
  *
  {
    pop => unit,   //EN Population start of year
    py => duration() //EN Average population in year
  } //EN Quantity
};
```

The following `//LABEL` statements provide a French language label for the table dimensions and expressions using their explicit short names `age`, `pop`, and `py`:

```
//LABEL(TotalPopulationByYear.age,FR) Âge intègre
//LABEL(TotalPopulationByYear.pop,FR) Population au début de l'année
//LABEL(TotalPopulationByYear.py,FR) Population moyenne pendant l'année
```

The expression dimension of a table is a special case. It uses the fixed name `expression_dimension` as the second part of the two-part name.

Continuing the above example, a French version of the expression dimension of the table is supplied by

```
//LABEL(T01_LifeExpectancy.expression_dimension,FR) Quantité
```

This string is displayed in the UI in the header of the table's expressions.

Here's another example showing `//LABEL` comments with explicit short names to provide French versions for the dimensions of the parameter `UnionDurationBaseline` in `RiskPaths`:

```
  double UnionDurationBaseline
    union_order => [UNION_ORDER]     //EN Union order
    union_dur   => [UNION_DURATION]; //EN Union duration
  //LABEL(UnionDurationBaseline.union_order,FR) Ordre d'union
  //LABEL(UnionDurationBaseline.union_dur,FR) Durée d'union
```

Here's an example showing `//LABEL` comments with explicit short names to provide the translation for the levels (enumerators) of the classification `UnionDurationBaseline` in `RiskPaths`:

```
classification UNION_ORDER        //EN Union order
{
  first => UO_FIRST,  //EN First union
  second => UO_SECOND //EN Second union
};
//LABEL(UNION_ORDER.first,FR) Première
//LABEL(UNION_ORDER.second,FR) Deuxième
```

[back to symbol labels]
[back to topic contents]

Symbol labels: `//LABEL` with two-part name (Modgen scheme)

The other method to specify the second part of a two-part name in a `//LABEL` comment uses the naming scheme used in Modgen for `//LABEL`.

In the Modgen naming scheme, a `LABEL` comment for a parameter dimension has the form
`//LABEL(ParameterName.DimN,LANG) text`
where `N` is `{0,1,...,rank-1}`, `rank` is the number of parameter dimensions, `LANG` is the language code, and `text` is the label.

and a `LABEL` comment for a table dimension has the form
`//LABEL(TableName.DimN,LANG) text`
where `N` is `{0,1,...,rank}`, `rank` is the number of classificatory dimensions in the table, `LANG` is the language code, and `text` is the label.
NB: The naming scheme for table dimensions differs from that used for default short names. Unlike a default short name, it *includes* the expression dimension of the table in the numbering.

In the Modgen naming scheme, a `LABEL` comment for a table expression has the form
`//LABEL(TableName.ExprN,LANG) text`
where `N` is `{0,1,...,expressions}`, `expressions` is the number of expressions in the table, `LANG` is the language code, and `text` is the label.

Here's the version of the French translation for the labels of the previous example using the Modgen naming scheme:

```
//LABEL(TotalPopulationByYear.Dim0,FR) Âge intègre
//LABEL(TotalPopulationByYear.Expr0,FR) Nombre total de cas simulés
//LABEL(TotalPopulationByYear.Expr1,FR) Durée totale
//LABEL(TotalPopulationByYear.Expr2,FR) Espérance de vie
```

[back to symbol labels]
[back to topic contents]

Symbol labels: `LABEL` text file

A symbol label can be supplied as a `.txt` file in the `doc` sub-directory of the model. For example, the file `RiskPaths/doc/LABEL.ProbMort.FR.txt` provides a French language version of the label of the `ProbMort` parameter. The file contents is a single line, which in this example is

```
Probabilités de décès
```

To be recognized as a `LABEL` text file, the filename must be named as in the example: `LABEL`, followed by the symbol name, followed by the code of a model language, followed by `txt`, with these parts separated by the `.` character.

If both `//LABEL` and a `LABEL` text file are present, the `LABEL` text file takes precedence.

A label supplied inline with the symbol declaration in model code takes precedence over a label supplied through a `LABEL` text file.

[back to symbol labels]
[back to topic contents]

Symbol notes

A model symbol can have an associated descriptive *note*. The note for a published symbol such as a parameter, table, or classification can be viewed in the UI in context by clicking the information icon. A symbol note may also be incorporated into Generated Model Documentation in the model's Symbol Reference.

The content of a note can be provided in one of two ways: through a `/*NOTE` comment in model code, or through a `.md` markdown file in the `doc` sub-directory of the model.

Model notes are authored using markdown, with some augmentations and limitations, as described in writing content.

This subtopic is organized into the following sections:

- Symbol notes: `/*NOTE` in model code
- Symbol notes: `NOTE` markdown file
- Symbol notes: The special `model` symbol
- Symbol notes: Modgen formatting indicators

[back to topic contents]

Symbol notes: `/*NOTE` in model code

The text for a note can be given in model code using a `NOTE` comment, For example, in `RiskPaths`,

```
/*NOTE(Person.FirstPregEvent, EN)
   The first pregnancy event. This is the main event of analysis and
   censors all future union events.
*/
```

in the `Fertility.mpp` model code module provides a note in the human language `EN` for the event `FirstPregEvent` in the `Person` entity.

The rules for two-part symbol names in `NOTE` comments are the same as those for `//LABEL` given in symbol labels.

To be recognized as a human-language note, there must be no white space between `/*` and `NOTE`. Subsequent white space is optional.

[back to symbol notes]
[back to topic contents]

Symbol notes: `NOTE` markdown file

A symbol note can be supplied as a `.md` markdown file in the `doc` sub-directory of the model. For example, the file `RiskPaths/doc/NOTE.ProbMort.EN.md` provides the English language version of the descriptive note for the `ProbMort` parameter.

The file content uses the markdown syntax, which for `NOTE.ProbMort.EN.md` is:

```
* The values in `ProbMort` have no effect if [`CanDie`](#CanDie) is `false`.
* The [GetLabel(Person::DeathEvent)](#DeathEvent) function in `RiskPaths`
 treats the values in `ProbMort` as though they are hazards.
 Arguably, the annual probabilities of death in `ProbMort` should be converted to hazards
 using the formula
   \(h_{x}=-\ln(1-q_{x})\).
 where
 \(h_{x}\) is the hazard of death and \(q_{x}\) is the annual mortality rate.
```

and, depending on browser capabilities and environment, might render like this in the `RiskPaths` Symbol Reference, towards the top of the topic, after the hierarchy and before the dimensions table:

See writing content for more information about note content and rendering.

[back to symbol notes]
[back to topic contents]

Symbol notes: The special `model` symbol

A note provided for the special symbol `model` will be displayed as introductory text in the model UI and in the model's generated Symbol Reference.

For example, the `Info.mpp` module in the `OzProj` model provides the following note for the `model` symbol:

```
/*NOTE(model, EN)
  A series of OzProj model versions accompany the on-line course "Practical Microsimulation".
  That course presents how to use Modgen to construct a microsimulation model
  to produce detailed demographic projections from a large heterogenous microdata population.

  The course content also teaches how to use Modgen through concrete self-contained examples.

  The model is based on data from the web site of the Australian Bureau of Statistics.
*/
```

Clicking the information icon for the `OzProj` model in the UI displays a pop-up like

If provided, the note for the `model` symbol is incorporated into the model's Symbol Reference, but this can be disabled using the `symref_model_symbol` option, as described in customizing the symbol reference. If the model has autonomous authored documentation with a home page, a note for the `model` symbol might be redundant in model documentation.

[back to symbol notes]
[back to topic contents]

Symbol notes: Modgen formatting indicators

Modgen formatting indicators in notes are described in the Modgen Developer's Guide in section "Formatting of symbol notes" on page 217. By default, the OpenM++ compiler identifies and converts Modgen formatting indicators to equivalent markdown when it encounters a note in model code in `/*NOTE`.

If a model uses markdown exclusively in notes in model code, this conversion can be disabled using the following statement:

```
options convert_modgen_note_syntax = off;
```

In practice, markdown and Modgen formatting indicators can co-exist in a `/*NOTE` comment without interfering.

Modgen formatting indicators are ignored and not converted in `NOTE` markdown files.

[back to symbol notes]
[back to topic contents]

Identifying missing symbol documentation

A common issue for model developers is to identify undocumented symbols, and, once identified, insert the missing documentation in the model source code.

The OpenM++ compiler supports a family of options to aid that process. Each member of the family targets a specific kind of missing

documentation. When an option is set to `on`, the compiler will generate a warning for each occurrence of missing documentation of that kind. The warning includes the model code file and line where the symbol was declared, except for missing translation warnings which instead give the location of the untranslated text. In an IDE like Visual Studio, double-clicking on the warning in the log window navigates immediately to that model source code location in the IDE editor.

By default these options are `off`. Multiple options can be turned `on` at the same time.

Here's an example to identify all published parameters in `RiskPaths` which have no descriptive note in the model's default language. Inserting the following line in `ompp_framework.ompp`

```
options missing_note_warning_published_parameter = on;
```

causes the compiler to emit warnings like:

```
1>../code/Unions.mpp(39): warning : missing note for published parameter 'AgeBaselineForm1'
1>../code/Fertility.mpp(21): warning : missing note for published parameter 'AgeBaselinePreg1'
1>../code/Mortality.mpp(25): warning : missing note for published parameter 'CanDie'
1>../code/Mortality.mpp(26): warning : missing note for published parameter 'ProbMort'
1>../code/Unions.mpp(45): warning : missing note for published parameter 'SeparationDurationBaseline'
1>../code/Unions.mpp(42): warning : missing note for published parameter 'UnionDurationBaseline'
1>../code/Fertility.mpp(24): warning : missing note for published parameter 'UnionStatusPreg1'
```

Here's an example which identifies all published symbols in `IDMM` which have a descriptive label or note in the default language, but whose translation is missing in one of the model's other languages. The source of `IDMM` was changed for this example to deliberately create missing translations.

Inserting the following lines in `ompp_framework.ompp`

```
options missing_translated_label_warning_published_any = on;
options missing_translated_note_warning_published_any = on;
```

causes the compiler to emit warnings like:

```
1>../code/HostCore.mpp(89): warning : missing 'FR' translated label for published symbol 'event_count'
1>../code/HostCore.mpp(82): warning : missing 'FR' translated note for published symbol 'NumberOfHosts'
```

For missing translation warnings, the warning code location is the location of the label or note in the default language, not the location of the symbol declaration. That's so the warning can be used to navigate to the text to be translated.

The missing translated note warning above gave the code location `HostCore.mpp(82)`.
Here's an extract of the code starting at that location (line 82):

```
/*NOTE(NumberOfHosts,EN)
This number does not change during the simulation
because there are no births, immigration, or deaths.
*/
```

Double-clicking the warning in an IDE navigates directly to that `NOTE` in the IDE editor. The associated parameter `NumberOfHosts` is declared elsewhere.

The following table lists the available options to emit warnings for missing symbol documentation, grouped by category. The Scope column shows what produces a warning for the given option. For example, the `missing_label_parameter` option produces a warning for a missing label for a parameter or for a parameter group. It does not produce a missing label warning for a parameter dimension. That's because the default label for a parameter dimension is copied from the label of the enumeration for the dimension, and is usually sufficient. The scope of each option is deliberately restricted to produce useful actionable warnings.

Option	Scope
Labels	
missing_label_warning_enumeration	classification, classification level, range, partition
missing_label_warning_parameter	parameter, parameter group
missing_label_warning_table	table, table expression, table group

Option	Scope
missing_label_warning_attribute	attribute
Labels - Published	
missing_label_warning_published_enumeration	as above, but only if published
missing_label_warning_published_parameter	as above, but only if published
missing_label_warning_published_table	as above, but only if published
missing_label_warning_published_attribute	as above, but only if published
Notes - Published	
missing_note_warning_published_parameter	published parameter
missing_note_warning_published_table	published table
missing_note_warning_published_attribute	published attribute
Translated Labels and Notes	
missing_translated_label_warning_any	any symbol with an explicit label provided in the model's default language
missing_translated_note_warning_any	any symbol with a note provided in the model's default language
Translated Labels and Notes - Published	
missing_translated_label_warning_published_any	as above, but only for published symbols
missing_translated_note_warning_published_any	as above, but only for published symbols

[back to topic contents]

Tables

This topic is under construction and/or revision.

Topic summary, two sentences max.

Related topics

- Entity Tables: Entity tables
- Derived Tables: Derived tables
- Model Output Expressions: Model Output Expressions

Topic contents

- Introduction and outline
- Run-level results
- Overriding run-level results

Introduction and outline

Under construction: more content to follow...

[back to topic contents]

Run-level results

OpenM++ computes the run-level value of a table cell by aggregating the values of the table cell over the subs/replicates/members of the run. The aggregation method depends on the kind of model (case-based or time-based), the `measures_method` option, the kind of table (entity or derived), and for entity tables the measure statistic (count, sum, etc.).

Under construction: more content to follow...

Model code computes the values of a derived table, so for derived tables, OpenM++ sets the run-level aggregated result to the average of the sub values.

[back to topic contents]

Overriding run-level results

The formula to compute run-level results can be specified explicitly for a table measure by providing an expression in model code, as illustrated by the following example:

```
//EXPR T00_Test.E0 OM_SUM(acc1) / OM_DIV_BY(acc2)
```

The syntax has three parts:

1. The introductory `//EXPR` single-line comment
2. The two-part name of a table measure
3. The expression used to compute the run-level value for that measure, in model output expression syntax using the table's accumulators.

Here's a worked example, using a derived table named `T00_Test`:

```
derived_table T00_Test
{
  {
    E0,  //EN quotient
    E1,  //EN numerator
    E2   //EN denominator
  }
};
```

This table has no enumeration dimensions, so consists of a single cell (not counting the measure dimension). The measure dimension has 3 measures/expressions: `E0`, `E1`, and `E2`. As with all derived tables, each measure has an implicit associated 'fake' accumulator which for this table are named `acc0`, `acc1`, and `acc2`.

In this example, model code populates the cells mechanically for each sub as follows:

- The cell `E1` is always set to `1.0`.
- The cell `E2` is set to the number of the sub `{0,1,2, ...}`.
- The cell `E0` is set to the quotient `E1 / E2`.

Here are the values of `T00_Test` for each sub for a 2-sub run:

Sub	E0	E1	E2
0	1.0	1.0	1.0
1	0.5	1.0	2.0

▶ Here's the model code which populates `T00_Test` for each sub:

Because `T00_Test` is a derived table, the run-level value is normally calculated using the `OM_AVG()` aggregation function, as follows:

Sub	E0	E1	E2
0	1.0	1.0	1.0
1	0.5	1.0	2.0
Run-level formula	`OM_AVG(acc0)`	`OM_AVG(acc1)`	`OM_AVG(acc2)`
Run-level value	0.75	1.0	1.5

In words, the default formula computes `E0` by averaging the ratios in `E0` over the subs.
The run result for `E0` is

```
E0 = ((1/1) + (1/2))/2
   = (1.0 + 0.5) / 2
   = 1.5 / 2
   = 0.75
```

If the formula to compute `E0` at the run-level is overridden by

```
//EXPR T00_Test.E0 OM_SUM(acc1) / OM_DIV_BY(acc2)
```

the calculation instead becomes

Sub	E0	E1	E2
0	1.0	1.0	1.0
1	0.5	1.0	2.0
Run-level formula	`OM_SUM(acc1) / OM_DIV_BY(acc2)`	`OM_AVG(acc1)`	`OM_AVG(acc2)`
Run-level value	0.66666	1.0	1.5

In words, the override formula computes `E0` by summing the numerators over the subs, summing the denominators over the subs, and then computing the quotient.
The run result for `E0` is

```
E0 = (1+1)/(1+2)
   = 2.0 / 3.0
   = 0.66666
```

[back to topic contents]

Test Models

This topic contains detailed information about the OpenM++ `test_models` utility. `test_models` builds, runs, and compares results for different versions of the same model, or for the same model built in different ways or on different platforms.

Topic contents

- Introduction and overview
- **Windows** Quick start
- **Linux or MacOS** Quick start
- Concepts
- Arguments and Options
- Control - Files
- Output - Report
- Output - Files
- Example 1: Preparing the default run
- Example 2: A new OpenM++ release
- Example 3: A model code change (*under construction*)
- Example 4: When results differ (*under construction*)
- Example 5: Exercising a model in Debug (*under construction*)
- Example 6: A performance comparison
- Example 7: Using event trace output (*under construction, some content present*)

Introduction and overview

The `test_models` utility builds, runs, and compares results for different versions of the same model, or for the same model built on different platforms. It can process multiple models in a single invocation and can compare OpenM++ and Modgen versions of cross-compatible models. It can play a useful role during incremental model development, and for quality assurance.

`test_models` is efficient enough that it can be used routinely in model development workflow cycles, even for large models (using a small sized Default simulation). even multiple times per hour or more during active model development.

When used routinely to assess the effects of incremental model changes, `test_models` can help identify bugs in model logic. It highlights when model changes have unexpected effects in model outputs, by comparing all cells of all output tables between a Current run and a Reference run.

`test_models` can be used to compare outputs of ompp model versions across platforms. Such differences are rare, and are typically caused by errors in model logic (often due to bad C++ code in models, eg use of uninitialized variables, or accessing memory beyond array limits using index variables with bad values).

`test_models` works with the standard model folder structure illustrated in the sample models in the OpenM++ distribution. It organizes results herarchically in a subfolder `test_models` of the model folder. `test_models` organizes comparisons in two dimensions. The first dimension is {Reference, Current}. A user can tell `test_models` to move Current results to Reference results to create a new point of reference for subsequent output comparisons. The second dimension of comparison is the platform ‘flavour’ {modgen, ompp-win, ompp-linux, ompp-macos}. Arguments to `test_models` determine where model results will be stored and which comparisons will be performed after `test_models` does the requested runs. For speed and ease of use, each model output table is converted to three forms: full precision, rounded precision, and a summary 'digest' of the file contents. The use of rounded precision eliminates most (but not all) spurious differences, eg differences at 6-7th decimal place. The table digest makes it possible for `test_models` to determine whether two versions of a table differ at very high speed, without comparing the two tables cell-by-cell.

`test_models` can compare results of Modgen and ompp versions of models, despite large underlying differences in output storage schema and technology. It reads the database of each and converts output tables into a common comparable csv format.

`test_models` can be very useful for cross-compatible model conversion. One can break cross-compatible model conversion into atomic steps and use `test_models` after each atomic step to verify that Modgen results have not changed, even before an OpenM++ version of the model is buildable. The comparisons relevant during cross-compatible model conversion are

1. Current(modgen) vs. Reference(modgen) - Modgen results should remain identical at each atomic change to model source, when build in Modgen.

2. Current(ompp) vs. Current(modgen) – After the model code is cross-compatible, ie when the ompp version can be built with no compiler errors, the model can be run in both versions and model results should be identical, because the model specifications are identical (same model source code and same Default scenario). Sometimes they are not, in which case the tables which differ can sometimes provide a clue.

When tracking down the cause of an unexpected difference in two runs, it can cometimes be useful to simplify the Default scenario and run `test_models` again, to see which area of the model might responsible for the difference. For example, if all tables differ, but when one turns off immigration in the Default scenario all differences disappear, one knows that the immigration code is somehow responsible. It can also be useful to turn on event tracking in the model with a small simulated population. If `test_models` indicates that the two event tracking files are identical, but table results are different, then the cause could be a Modgen tabulation bug.

`test_models` is also useful for model development. One can break model development into small atomic steps to help identify unexpected changes in model outputs. If possible, one can structure model changes so that Default parameters and code changes should have no effect on outputs, eg by turning a new model option off in Default parameters. That can identify whether the new model code, when disabled using the new parameters, has unexpected effects on other parts of the model (it should not). The `test_models` comparisons typically used for model development are

1. Current(ompp) vs. Reference(ompp) - Model results from incremental changes should affect only new tables or expected interactions among model components. Once verified, one can tell test_models to copy Current results to Reference results (using the `--newref` option) to create a new Reference for the next atomic set of changes to model code.

If one prefers to work in the Modgen environment for model development, one can instead use

1. Current(modgen) vs. Reference(modgen) - See notes above.

2. Current(modgen) vs. Current(ompp) – From time to time, build the ompp version to identify if non cross-compatible code has crept into the model source, and rectify if so.

`test_models` also standardizes the detailed event trace outputs (if activated) in ompp and Modgen models so that they are comparable and readable. This can identify precisely when a simulation diverges in the Modgen and ompp versions of a model, or in (for example) two ompp versions of a model. This is useful for unexpected and hard-to-understand differences in simulations. If the time and event of earliest divergence in the two runs is insufficient to understand the cause in and of itself, that information provides what's required to set conditional breakpoints for a parallel debugging session of the two runs, stepping through the simulation in each of the two debugging sessions, to identify the precise code location responsible for the divergence in the simulation in the two versions.

`test_models` also notes some tombstone information about each set of outputs, eg the modgen version. It also keeps a copy of the build log output and Default run log output. `test_models` can also report the elpased time of various steps, including model build and model run. A count of compiler warnings is also reported.

`test_models` can run and process a single model, or multiple models in subfolders of a parent folder. That can be useful for testing mutliple scenarios of a single model, using mutliple git clones inside a parent folder. It can also be useful to bulk test multiple models routinely, for example after a change in the OpenM++ version.

[back to topic contents]

Windows Quick Start

1. Verify installation of `test_models` (Windows)

A 64-bit Windows executable version of `test_models` is distributed with OpenM++ at `OM_ROOT/bin/test_models.exe`, where `OM_ROOT` stands for the OpenM++ installation directory. A 32-bit version is at `OM_ROOT/bin/test_models32.exe`. The 32-bit version may be required for the Modgen flavour to work successfully using the executable version of `test_models`. To test installation and operation of `test_models`, open a command prompt, change the current directory to `OM_ROOT/bin`, and type the command

```
test_models -v
```

Output should be similar to the following:

`test_models version 2.1`

`test_models` is written in the Perl language, and distributed with OpenM++ at `OM_ROOT/Perl/test_models.pl`. Most examples in this topic invoke `test_models` using the Perl interpreter from the `OM_ROOT/Perl` directory, eg

`perl test_models.pl -v`

On Windows, unless you have Perl and the required Perl components installed, invoke the executable version of `test_models` from the `OM_ROOT/bin` directory with a command like

`test_models -v`

or

`test_models32 -v`

2. Display `test_models` options (Windows)

From the `OM_ROOT/bin` directory, type the command

`test_models -h`

Output should be similar to the following:

```
test_models [-hmv] [long options...] model...
        -m STR --models_root STR    directory containing models (default is .)
        --newref                    replace Reference results with Current
                                    results
        --noompp                    skip OpenM++ build and run
        --nomodgen                  skip Modgen build and run
        --nocomp                    skip flavour comparison
        --allfiles                  report all different and orphaned files
        --timing                    report elapsed time of steps
        --nosteps                   skip reporting which step is being
                                    performed
        --config STR                build configuration: debug or
                                    release(default)
        --mpi_processes INT         build MPI version and run with n
                                    processes (default 0, means no MPI)
        --gencode                   keep a copy of the generated C++ code
        --ini STR                   OpenM++ model ini file to pass to model
                                    (in model root, default is
                                    test_models.ini if present)
        --clean                     remove all build files after run
        --significant_digits INT    significant digits (default 6)
        --nocells                   disable fallback cell-by-cell
                                    verification of differing tables and copy
                                    of original data
        -h --help                   report usage message and exit
        -v --version                report test_models version and exit
        --windows_platform STR      Windows platform: x64(default) or Win32
        --modgen_platform STR       Modgen platform: Win32(default) or x64on of unrounded versions
                                    of tables
```

3. Run `test_models` on the RiskPaths model (Windows)

From the `OM_ROOT/bin` directory, type the command

`test_models -m ../models RiskPaths`

If Modgen is installed, use `test_models32` instead of `test_models`. `test_models` will build and run both the OpenM++ and the Modgen versions of RiskPaths, and compare their results. Output should be similar to the following:

```
        ========================
         test_models 2.0
        ========================

        Testing: RiskPaths
        modgen settings: version=12,1,3,0 (2019-12-19 20:31 GMT) platform=Win32 configuration=release
        ompp-win settings: compiler=omc.exe (2021-05-28 02:28 GMT) platform=x64 configuration=release

        RiskPaths: modgen: Build model and prepare Default scenario
        RiskPaths: modgen: Run model using RiskPaths.ini
        RiskPaths: modgen: Convert outputs (7 digits of precision)
        RiskPaths: modgen: Create digests of current outputs
        RiskPaths: modgen: No Reference outputs - create using Current outputs
        RiskPaths: modgen: Current vs. Reference:   9 the same (of 9)
        RiskPaths:
        RiskPaths: ompp-win: Build and publish model and Default scenario
warning => RiskPaths: ompp-win: 10 build warning(s) - see RiskPaths/test_models/current/ompp-win/logs/build.log
        RiskPaths: ompp-win: Run model using RiskPaths.ini
        RiskPaths: ompp-win: Convert outputs (7 digits of precision)
        RiskPaths: ompp-win: Create digests of current outputs
        RiskPaths: ompp-win: No Reference outputs - create using Current outputs
        RiskPaths: ompp-win: Current vs. Reference:   9 the same (of 9)
        RiskPaths:
        RiskPaths:     Flavour comparisons:
        RiskPaths:
        RiskPaths: ompp-win vs. modgen:     Reference:   9 the same (of 9)
        RiskPaths:
        RiskPaths: ompp-win vs. modgen:     Current:     9 the same (of 9)
```

If Modgen is not installed, output will not include the Modgen portion nor the flavour comparison portions of the report.

`test_models` uses the Microsoft application `msbuild.exe` to build models. `msbuild.exe` is normally installed as part of Visual Studio installation, but the install location can vary. `test_models` attempts to determine the location of `msbuild.exe` but may not always succeed. If you encounter issues running `test_models` which seem related to `msbuild.exe` you can provide `test_models` the location explicitly using the environment variable `MSBUILD_EXE`, for example:

```
set MSBUILD_EXE=C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\MSBuild\Current\Bin\msbuild.exe
```

in the command window before invoking `test_models`.

[back to topic contents]

Linux or MacOS Quick Start

1. Verify installation of `test_models` (Linux, MacOS)

`test_models` is a Perl script distributed with OpenM++ at `OM_ROOT/Perl/test_models.pl`, where `OM_ROOT` stands for the OpenM++ installation directory. To test installation and operation of `test_models`, open a command prompt, change the current directory to `OM_ROOT\Perl`, and type the command

```
perl test_models.pl -v
```

Output should be similar to the following:

```
test_models version 2.1
```

Depending on your operating system version and installation history, Perl may ask you to install missing Perl modules required by `test_models.pl`. If so, it will name them explicitly when you invoke `test_models.pl`. We do recommend to use `cpanm` for Perl modules installation. Typical scenario is:

```
cpan App::cpanminus            # initialize cpanm, if not done before

cpanm Getopt::Long::Descriptive
cpanm Capture::Tiny
cpanm Config::Tiny
cpanm File::Copy::Recursive
cpanm File::Which
```

Above list of modules can be different and depends on your current Perl configuration, and on the version of test_models.

2. Display `test_models` options (Linux, MacOS)

From the `OM_ROOT/Perl` directory, type the command

`perl test_models.pl -h`

Output should be similar to the following:

```
test_models [-hmv] [long options...] model...
        -m STR --models_root STR    directory containing models (default is .)
        --newref                    replace Reference results with Current
                                    results
        --noompp                    skip OpenM++ build and run
        --nomodgen                  skip Modgen build and run
        --nocomp                    skip flavour comparison
        --allfiles                  report all different and orphaned files
        --timing                    report elapsed time of steps
        --nosteps                   skip reporting which step is being
                                    performed
        --config STR                build configuration: debug or
                                    release(default)
        --mpi_processes INT         build MPI version and run with n
                                    processes (default 0, means no MPI)
        --gencode                   keep a copy of the generated C++ code
        --ini STR                   OpenM++ model ini file to pass to model
                                    (in model root, default is
                                    test_models.ini if present)
        --clean                     remove all build files after run
        --significant_digits INT    significant digits (default 6)
        --nocells                   disable fallback cell-by-cell
                                    verification of differing tables and copy
                                    of original data
        -h --help                   report usage message and exit
        -v --version                report test_models version and exit
        --windows_platform STR      Windows platform: x64(default) or Win32
        --modgen_platform STR       Modgen platform: Win32(default) or x64
```

3. Run `test_models` on the RiskPaths model (Linux, MacOS)

From the `OM_ROOT/Perl` directory, type the command

`perl test_models.pl -m ../models RiskPaths`

Output should be similar to the following:

```
        ========================
         test_models 2.0
        ========================

        Testing: RiskPaths
        ompp-linux settings: compiler=omc (2021-05-29 16:47 GMT) configuration=release

        RiskPaths: ompp-linux: Build and publish model and Default scenario
        RiskPaths: ompp-linux: Run model using RiskPaths.ini
        RiskPaths: ompp-linux: Convert outputs (7 digits of precision)
        RiskPaths: ompp-linux: Create digests of current outputs
        RiskPaths: ompp-linux: No Reference outputs - create using Current outputs.
        RiskPaths: ompp-linux: Current vs. Reference:   9 the same (of 9)
```

[back to topic contents]

Concepts

This subtopic describes key underlying concepts used in `test_models`.

- Run version: Current or Reference
- Run flavour: Windows, Linux, MacOS, Modgen
- Output comparison:
- Build options:
- Run options:

[back to topic contents]

Run version

=== under construction ===

- Current or Reference

[back to concepts]
[back to topic contents]

Run flavour

=== under construction ===

- Windows, Linux, MacOS, Modgen

[back to concepts]
[back to topic contents]

Output comparison

=== under construction ===

[back to concepts]
[back to topic contents]

Build options

=== under construction ===

[back to concepts]
[back to topic contents]

Run options

=== under construction ===

[back to concepts]
[back to topic contents]

Arguments and Options

This subtopic describes the command line options and arguments of `test_models`, organized into sections. It also describes default `test_models` behaviour for each section.

- Syntax: The syntax of test_models arguments
- Models: The models to process
- Actions: The actions to perform on a model
- Verbosity: The level of report detail
- Build: How a model is built
- Run: How a model is run
- Comparison: How result are compared
- Informational: Informational options
- EventTrace: EventTrace options

A complete list of options is displayed by issuing the command

`perl test_models.pl -h`

```
test_models [-hmv] [long options...] model...
        -m STR --models_root STR    directory containing models (default is .)
        --newref                    replace Reference results with Current
                                    results
        --noompp                    skip OpenM++ build and run
        --nomodgen                  skip Modgen build and run
        --nocomp                    skip flavour comparison
        --allfiles                  report all different and orphaned files
        --timing                    report elapsed time of steps
        --nosteps                   skip reporting which step is being
                                    performed
        --config STR                build configuration: debug or
                                    release(default)
        --mpi_processes INT         build MPI version and run with n
                                    processes (default 0, means no MPI)
        --gencode                   keep a copy of the generated C++ code
        --ini STR                   OpenM++ model ini file to pass to model
                                    (in model root, default is
                                    test_models.ini if present)
        --clean                     remove all build files after run
        --significant_digits INT    significant digits (default 6)
        --zero_fuzz NUM             zero fuzz value (default 1e-15)
        --nocells                   disable fallback cell-by-cell
                                    verification of differing tables and copy
                                    of original data
        -h --help                   report usage message and exit
        -v --version                report test_models version and exit
        --time_format STR           time format for event trace (default
                                    13.6f)
        --modgen_id_offset INT      offset adjustment to Modgen IDs (default
                                    0)
        --windows_platform STR      Windows platform: x64(default) or Win32
        --modgen_platform STR       Modgen platform: Win32(default) or x64
```

[back to arguments and options]
[back to topic contents]

Syntax

Arguments to `test_models` consist of a series of options followed by a list of models to process. Either or both may be empty, in which case default values are used. An option starts with `--` followed immediately by the option name. Some options may be followed by an option value. Some options have a synonymous short form consisting of a single `-` followed imediately by a single letter.

[back to arguments and options]
[back to topic contents]

Models

The `--models_root` option (short form `-m`) specifies the parent folder of one or more folders each of which contains a model. If not specified, the models root is the current working directory. Consider the following command, issued after setting the current working directory to `OM_ROOT/Perl`:

```
perl test_models.pl -m ../models NewCaseBased RiskPaths
```

On Windows without Perl installed, a similar command would be issued after setting the current working directory to `OM_ROOT/bin`:

```
test_models -m ../models NewCaseBased RiskPaths
```

The command instructs `test_models` to set the models root to the sister directory `models` in the OpenM++ distribution. That folder is `OM_ROOT/models` and contains several sample models distributed with OpenM++, each in its own subfolder, eg `OM_ROOT/models/RiskPaths`. The final two arguments instruct `test_models` to process the two models in the subfolders `OM_ROOT/models/NewCaseBased` and `OM_ROOT/models/RiskPaths`.

On Linux, output is similar to:

```
        ========================
         test_models 2.0
        ========================

        Testing: NewCaseBased, RiskPaths
        ompp-linux settings: compiler=omc (2021-05-29 16:47 GMT) configuration=release

        NewCaseBased: ompp-linux: Build and publish model and Default scenario
        NewCaseBased: ompp-linux: Run model using test_models.ini
        NewCaseBased: ompp-linux: Convert outputs (7 digits of precision)
        NewCaseBased: ompp-linux: Create digests of current outputs
        NewCaseBased: ompp-linux: No Reference outputs - create using Current outputs
        NewCaseBased: ompp-linux: Current vs. Reference:   3 the same (of 3)

        RiskPaths: ompp-linux: Build and publish model and Default scenario
        RiskPaths: ompp-linux: Run model using RiskPaths.ini
        RiskPaths: ompp-linux: Convert outputs (7 digits of precision)
        RiskPaths: ompp-linux: Create digests of current outputs
        RiskPaths: ompp-linux: No Reference outputs - create using Current outputs
        RiskPaths: ompp-linux: Current vs. Reference:   9 the same (of 9)
```

On Windows with Modgen installed, output is similar to:

```
        ========================
         test_models 2.0
        ========================

        Testing: NewCaseBased, RiskPaths
        modgen settings: version=12,1,3,0 (2019-12-19 20:31 GMT) platform=Win32 configuration=release
        ompp-win settings: compiler=omc.exe (2021-05-28 02:28 GMT) platform=x64 configuration=release

        NewCaseBased: modgen: Build model and prepare Default scenario
        NewCaseBased: modgen: Run model using test_models.ini
        NewCaseBased: modgen: Convert outputs (7 digits of precision)
        NewCaseBased: modgen: Create digests of current outputs
        NewCaseBased: modgen: No Reference outputs - create using Current outputs
        NewCaseBased: modgen: Current vs. Reference:   3 the same (of 3)
        NewCaseBased:
        NewCaseBased: ompp-win: Build and publish model and Default scenario
warning => NewCaseBased: ompp-win: 3 build warning(s) - see NewCaseBased/test_models/current/ompp-win/logs/build.log
        NewCaseBased: ompp-win: Run model using test_models.ini
        NewCaseBased: ompp-win: Convert outputs (7 digits of precision)
        NewCaseBased: ompp-win: Create digests of current outputs
        NewCaseBased: ompp-win: No Reference outputs - create using Current outputs
        NewCaseBased: ompp-win: Current vs. Reference:   3 the same (of 3)
        NewCaseBased:
        NewCaseBased:     Flavour comparisons:
        NewCaseBased:
        NewCaseBased: ompp-win vs. modgen:      Reference:   3 the same (of 3)
        NewCaseBased:
        NewCaseBased: ompp-win vs. modgen:      Current:     3 the same (of 3)

        RiskPaths: modgen: Build model and prepare Default scenario
        RiskPaths: modgen: Run model using RiskPaths.ini
        RiskPaths: modgen: Convert outputs (7 digits of precision)
        RiskPaths: modgen: Create digests of current outputs
        RiskPaths: modgen: No Reference outputs - create using Current outputs
        RiskPaths: modgen: Current vs. Reference:   9 the same (of 9)
        RiskPaths:
        RiskPaths: ompp-win: Build and publish model and Default scenario
warning => RiskPaths: ompp-win: 10 build warning(s) - see RiskPaths/test_models/current/ompp-win/logs/build.log
        RiskPaths: ompp-win: Run model using RiskPaths.ini
        RiskPaths: ompp-win: Convert outputs (7 digits of precision)
        RiskPaths: ompp-win: Create digests of current outputs
        RiskPaths: ompp-win: No Reference outputs - create using Current outputs
        RiskPaths: ompp-win: Current vs. Reference:   9 the same (of 9)
        RiskPaths:
        RiskPaths:     Flavour comparisons:
        RiskPaths:
        RiskPaths: ompp-win vs. modgen:      Reference:   9 the same (of 9)
        RiskPaths:
        RiskPaths: ompp-win vs. modgen:      Current:     9 the same (of 9)
```

If no models are specified on the command line, `test_models` will process **All** subfolders of the models root.

[back to arguments and options]
[back to topic contents]

Actions

Unless instructed otherwise, `test_models` builds and runs all available 'flavours' of a model on the current operating system. On Windows, two

flavours are available: OpenM++ and Modgen (if installed). On Linux and MacOS, the only available flavour is OpenM++. After building and running each flavour, `test_models` compares Current results to existing Reference results. If there are no Reference results, `test_models` creates them by copying Current results. After completing all flavours for a model, `test_models` compares results for other flavours (if present) to results for the OpenM++ flavour on the current operating system.

This default behaviour can be modified by the following command options:

- `--newref` For each flavour in this invocation, discard all Reference results and replace them with Current results
- `--noompp` Suppress processing the OpenM++ flavour
- `--nomodgen` Suppress processing the Modgen flavour (only applies if invoked on Windows)
- `--nocomp` Suppress reporting on differences between flavours

For example, on Windows, the command

`test_models -m ../models --noompp --nomodgen RiskPaths`

would process the RiskPaths model, but skip build and run of both Windows flavours (OpenM++ and Modgen). `test_models` would still compare results for any previously run flavours to OpenM++ Windows results (if present). Output might be similar to the following (or could be empty, if no other flavours were run previously):

```
========================
 test_models 2.0
========================

Testing: RiskPaths

RiskPaths:
RiskPaths:    Flavour comparisons:
RiskPaths:
RiskPaths: ompp-win vs. modgen:     Reference:   9 the same (of 9)
RiskPaths:
RiskPaths: ompp-win vs. modgen:     Current:     9 the same (of 9)
RiskPaths:
RiskPaths: ompp-win vs. ompp-linux:  Reference:   9 the same (of 9)
RiskPaths:
RiskPaths: ompp-win vs. ompp-linux:  Current:     9 the same (of 9)
```

In this invocation, the RiskPaths model was neither built nor run. `test_models` detected the presence of Current and Reference results for two other flavours (Modgen and OpenM++ on Linux), and compared them to results from OpenM++ on Windows which were present from a previous invocation of `test_models`. Incidentally, because `test_models` uses digests to compare results, comparing all results between two flavours is almost instantaneous, even for models with many tables or with very large tables.

[back to arguments and options]
[back to topic contents]

Verbosity

Unless instructed otherwise, `test_models` reports which step it is processing, It does not report the elapsed time of each step. When comparing results, `test_models` reports file counts and reports by name the first five differing or orphaned files.

This default behaviour can be modified by the following command options:

- `--allfiles` Report each differing and orphaned file by name, not just the first five
- `--timing` Also report the elapsed time for each step
- `--nosteps` Don't report which step is being processed

For example, on Windows, the command

`perl test_models.pl -m ../models --nomodgen --nocomp --timing --allfiles --significant_digits 8 OzProj`

instructs `test_models` to build and run just the OpenM++ version of OzProj, suppress the flavour comparison output, report elapsed time of each step, and report every differing file by name, not just the first five. For illustrative purposes, this invocation also changes the number of significant digits used to construct digests from 7 to 8 to deliberately produce differences between Current and Reference OzProj results. Output is similar to the following:

```
        ========================
         test_models 2.0
        ========================

        Testing: OzProj
        ompp-win settings: compiler=omc.exe (2021-05-28 02:28 GMT) platform=x64 configuration=release

        OzProj: ompp-win: Build and publish model and Default scenario
        OzProj: ompp-win: Build time 0m 7s
 warning => OzProj: ompp-win: 63 build warning(s) - see OzProj/test_models/current/ompp-win/logs/build.log
        OzProj: ompp-win: Run model using OzProj.ini
        OzProj: ompp-win: Run time 0m 2s
        OzProj: ompp-win: Convert outputs (8 digits of precision)
        OzProj: ompp-win: Convert time 0m 0s
        OzProj: ompp-win: Create digests of current outputs
        OzProj: ompp-win: Digest time 0m 0s
        OzProj: ompp-win: Current vs. Reference:  13 the same (of 21)
DIFFERS ===> OzProj: ompp-win: Current vs. Reference:   8 differ
DIFFERS ===> OzProj: ompp-win: Current vs. Reference: DIFFERS:        0_SIGNIFICANT_DIGITS.txt
DIFFERS ===> OzProj: ompp-win: Current vs. Reference: DIFFERS:        Experiment1.csv
DIFFERS ===> OzProj: ompp-win: Current vs. Reference: DIFFERS:        Experiment2.csv
DIFFERS ===> OzProj: ompp-win: Current vs. Reference: DIFFERS:        Experiment3.csv
DIFFERS ===> OzProj: ompp-win: Current vs. Reference: DIFFERS:        Experiment4.csv
DIFFERS ===> OzProj: ompp-win: Current vs. Reference: DIFFERS:        Experiment5.csv
DIFFERS ===> OzProj: ompp-win: Current vs. Reference: DIFFERS:        Experiment6.csv
DIFFERS ===> OzProj: ompp-win: Current vs. Reference: DIFFERS:        PersonYearsLived.csv
```

The specially-named 'result' file `0_SIGNIFICANT_DIGITS.txt` is described elswhere in this topic.

[back to arguments and options]
[back to topic contents]

Build

Unless instructed otherwise, `test_models` builds a Release version of a model, without MPI multi-processing capability, and does not save copy of generated C++ code.

This default behaviour can be modified by the following command options:

- `--config` Build configuration: debug or release(default)
- `--mpi_processes` Build MPI version (default 0, means no MPI)
- `--gencode` Save a copy of the generated C++ code in subfolder `generated_code`

The Debug version of a model is typically considerably slower than the Release version. The Debug version can, however, perform run-time checks which are absent in the Release version. This is particularly true if C++ model code uses assert↗ to verify that expected logical conditions are satisfied at run time (in Debug mode only).

Supply a non-zero value to `--mpi_processes` to build an MPI-enabled model (OpenM++ only). The value will be used to launch that number of MPI instances when the model is subsequently run as described in Run. The suffix `_mpi` will be appended to the name of the model executable to distinguish it from the normal non-MPI version.

The specialized --gencode option creates a copy of the C++ code generated by the omc compiler (or by the Modgen compiler for the `modgen` flavour). If this option is activated, the `src` temporary build folder is copied to the output folder `generated_code`. The `--gencode` option can be used to compare the C++ code generated by different versions of the OpenM++ compiler.

Incidentally, models are capable of multi-threading independent of MPI.

[back to arguments and options]
[back to topic contents]

Run

Unless instructed otherwise, `test_models` builds and runs a non-MPI enabled model using the Default scenario. It retains all files from the build, including the model executable and database after the model is run. Also by default, `test_models` looks in the model root for a model ini file to use to run the model. The search for the ini file proceeds as follows: If the file `test_models.ini` is present, it is used. If `test_models.ini` is not present, the file `MODEL.ini` is used if present, where `MODEL` stands for the name of the model. For more on how `test_models` uses a model ini file, see Control - Files.

This default behaviour can be modified by the following command options:

- `--ini` The name of the OpenM++ model ini used when the model is run
- `--mpi_processes` The number of mpi processes to launch
- `--clean` Remove all build files after the run

For example, if the file `OM_ROOT/RiskPaths/Test.ini` exists with content

```
[OpenM]
SubValues = 16
Threads = 4

[Parameter]
SimulationCases = 16000000
```

and the following command is issued

`perl test_models.pl -m ../models --newref --nomodgen --nocomp --timing --mpi_processes 4 --ini Test.ini RiskPaths`

an MPI-enabled version of RiskPaths is built and 4 instances are launched on the workstation under MPI control. The run consists of 16 replicates with 1,000,000 cases in each replicate, for a total of 16,000,000 cases. Each MPI instance will run with 4 threads.

Output is similar to the following:

```
        ========================
         test_models 2.0
        ========================

        Testing: RiskPaths
        ompp-win settings: compiler=omc.exe (2021-05-28 02:28 GMT) platform=x64 configuration=release mpi_processes=4

        RiskPaths: ompp-win: Deleting previous Reference information
        RiskPaths: ompp-win: Build and publish model and Default scenario
        RiskPaths: ompp-win: Build time 0m 5s
warning => RiskPaths: ompp-win: 10 build warning(s) - see RiskPaths/test_models/current/ompp-win/logs/build.log
        RiskPaths: ompp-win: Run model using Test.ini
        RiskPaths: ompp-win: Run time 2m 15s
        RiskPaths: ompp-win: Convert outputs (7 digits of precision)
        RiskPaths: ompp-win: Convert time 0m 0s
        RiskPaths: ompp-win: Create digests of current outputs
        RiskPaths: ompp-win: Digest time 0m 0s
        RiskPaths: ompp-win: No Reference outputs - create using Current outputs
        RiskPaths: ompp-win: Current vs. Reference:   9 the same (of 9)
```

The truncated log file of the run is similar to the following:

```
2021-05-31 16:18:57.819 RiskPaths
2021-05-31 16:18:57.819 RiskPaths
2021-05-31 16:18:57.993 Parallel run of 4 modeling processes, 4 thread(s) each
2021-05-31 16:18:57.994 Model build    : Windows 64 bit Release
2021-05-31 16:18:57.995 Prepare fixed and missing parameters
2021-05-31 16:18:58.012 Run: 102 Default
2021-05-31 16:18:58.012 Get scenario parameters for process
2021-05-31 16:18:58.013 Model build    : Windows 64 bit Release
2021-05-31 16:18:58.013 Prepare fixed and missing parameters
2021-05-31 16:18:58.013 Run: 102 Default
2021-05-31 16:18:58.014 Get scenario parameters for process
2021-05-31 16:18:58.014 member=0 Bind scenario parameters
2021-05-31 16:18:58.014 member=0 Compute derived parameters
2021-05-31 16:18:58.014 member=0 Prepare for simulation
2021-05-31 16:18:58.015 member=1 Bind scenario parameters
2021-05-31 16:18:58.015 member=1 Compute derived parameters
2021-05-31 16:18:58.015 member=1 Prepare for simulation
2021-05-31 16:18:58.015 member=0 Simulation progress=0% cases=0
...
```

[back to arguments and options]
[back to topic contents]

Comparison

After running a model, `test_models` extracts each output table from the model database and saves a copy of the original data as well as a version rounded to 6 significant digits. `test_models` then computes and stores a digest of each rounded csv file which it uses for subsequent comparisons. When a rounded table differs, `test_models` verifies that at least one cell differs by more than one part per million using the original table data, before reporting the table as different.

This default behaviour can be modified by the following command options:

- `--significant_digits` The number of significant digits for table result comparison
- `--zero_fuzz` Set table cell values to zero if within this value of zero
- `--nocells` Disable fallback cell-by-cell verification of differing tables and copy of original data

The number of significant digits used for rounding can be modified using the `--significant_digits` option. The default value is 6 (one part in one million). This level of precision can help reduce the number of differing tables reported by test_models while still identifying differences of substantive or logical significance.

Rounding reduces the number of spurious differences which would otherwise be reported by `test_models`. It is not unusual for different model versions to produce slightly different results, often at the level of numerical precision (15-16 digits of precision). This can occur due to different C++ compiler optimizations, or by logically equivalent but slightly different algorithms in models. Rounding to less precision reduces the number of spurious differences reported by `test_models`.

Most spurious differences are eliminated by rounding, but not all. For example, two numbers can differ only at the 7th digit of precision but round to different values at 6 digits of precision if the two numbers happen to fall on different sides of a 'rounding boundary' (last digit 5). `test_models` eliminates these false positives by verifying differing rounded tables using the original unrounded data. In practice, this verification is rapid. This verification can be disabled using the --nocells command line option. The --nocells option also disables the creation of original (unrounded) copies of model output tables.

The value of a table cell can be undefined, eg the mean value of an attribute for a cell with no observations. This is not the same as a table cell value of zero, eg a mean value of 0.0 calculated from one or more observations. `test_models` considers an undefined value to be different from the value 0.

Table measures using subtraction can sometimes produce values very close to zero as an artifact of floating-point computations. The presence of such values can produce false positives: tables which `test_models` considers different but which are not different substantively. `test_models` uses a `zero-fuzz` value to handle this situation. If the absolute value of a table cell is within `zero_fuzz` of `0.0`, the table cell will be set to `0.0`. The default `zero_fuzz` value is `1e-15`. The `zero_fuzz` value can be set to 0 to disable `zero_fuzz` altogether.

`test_models` computes digests using the MD5 cryptographic algorithm. For example, the MD5 digest of the RiskPaths table `T04_FertilityRatesByAgeGroup` in the `Default` run, rounded to 6 digits and normalized to csv is the hexadecimal value `993ead71da6eed5dde398342278f629a`. The MD5 digest is always 32 hexadecimal digits in length, independent of the size of the input file. Files with different digests are guaranteed to be different, and it is extremely unlikely (read almost impossible) for different files to have the same digest.

[back to arguments and options]
[back to topic contents]

Informational

The following options are informational.

- `--help` report usage message and exit
- `--version` report test_models version and exit

They have short form versions `-h` and `-v`.

If either option is on the command line, `test_models` will display the requested information and immediately exit.

[back to arguments and options]
[back to topic contents]

EventTrace

The following options affect the normalization of event trace output, if present..

- `--time_format STR` time format for event trace (default 13.6f)
- `--modgen_id_offset INT` offset adjustment to Modgen IDs (default 0)

The `--time_format` option changes the precision of time in the normalized event trace report. For example, specifying `--time_format 19.12f` increases the precision from 6 to 12 digits after the decimal point, producing output like

```
1969.960131010721 BirthdayEvent (Person 29)
        1970.960131010721 timeBirthdayEvent (Person 29)
```

The `--modgen_id_offset` option corrects the entity ID in the Modgen normalized event trace report. This is helpful if the run reproduces a single case from a larger run using a case seed. For such runs, Modgen assigns entity IDs incorrectly, causing spurious differences with ompp event trace output. To fix the issue, the entity IDs in the Modgen event trace report need to be offset by the number of the sub they came from. That sub # may already be known, but if not it can be calculated from the case seed by dividing the case seed by 2^31 and taking the integer part of the result. For example, to determine the sub # of the case seed `11636581014`, divide it by `2147483648` giving `5.4187052948...`. The sub # of the case is `5`. Specifying `--modgen_id_offset 5` will offset all entity IDs in the Modgen event trace report by 5, correcting the error and bringing the Modgen event trace report into alignment.

[back to arguments and options]
[back to topic contents]

Deprecated

- `--windows_platform` Windows platform: x64(default) or Win32
- `--modgen_platform` Modgen platform: Win32(default) or x64

If desired, a 32-bit OpenM++ version of a model can be built instead of a 64-bit version, on Windows only, using the `--windows_platform` option.

Possibly, a 64-bit version of a Modgen model can be built instead of a 32-bit version using the `--modgen_platform` option. The build or run may fail, and run functionality may be limited or compromised.

[back to arguments and options]
[back to topic contents]

Control - Files

=== under construction ===

`test_model` operation is controlled mainly by command-line options, but also by the presence and contents of optional specially-named files in the model folder.

OpenM++ model ini files are described here. `test_models` always enables Model development options by automatically specifying the `-OpenM.IniAnyKey` to the model when a model ini file is used.

[back to topic contents]

Output - Report

=== under construction ===

`test_models` writes a report line by line to the command window where it was invoked as it carries out operations.

[back to topic contents]

Output - Files

`test_models` produces two kinds of output: A report it writes to the command window where it was invoked, and files which record and persist information about the model runs it performs. The files created by `test_folders` can be useful to probe details behind the summary information in a `test_models` report. For example, `test_models` reports a count of warnings during model build and also creates the file `build.log` containing more information about those warnings.

This subtopic contains the following sections.

- test_models folder: The test_models folder hierarchy and contents
- table outputs: The tables from a model run
- event trace: The normalized event trace from a model run

test_models folder

The output files generated by `test_models` are in the subfolder `test_models` of the model folder. The `test_models` subfolder and hierarchy are created automatically by `test_models` as needed. The `test_models` subfolder can be deleted at any time, and will be recreated on a subsequent invocation of

`test_models`.

The following diagram shows the hierarchical structure of the `test_models` folder for the `RiskPaths` model, with some other subfolders of `RiskPaths` shown for context. In this example, `test_models` was previously invoked for the three flavours `ompp-win`, `ompp-linux`, and `modgen`.

```
RiskPaths
├───code
├───modgen
│   ├───bin
│   └───src
├───ompp
│   ├───bin
│   └───src
├───ompp-linux
│   ├───bin
│   └───src
├───parameters
│   └───Default
└───test_models
    ├───current
    │   ├───modgen
    │   │   ├───logs
    │   │   └───outputs
    │   │       └───original
    │   ├───ompp-linux
    │   │   ├───logs
    │   │   └───outputs
    │   │       └───original
    │   └───ompp-win
    │       ├───logs
    │       └───outputs
    │           └───original
    └───reference
        ├───modgen
        │   ├───logs
        │   └───outputs
        │       └───original
        ├───ompp-linux
        │   ├───logs
        │   └───outputs
        │       └───original
        └───ompp-win
            ├───logs
            └───outputs
                └───original
```

`test_models` organizes information using two folder hierarchy levels, as illustrated. For example, `RiskPaths` results for the Reference run with the `ompp-win` flavour are in the folder `RiskPaths/test_models/reference/ompp-win`.

Below is a list of all files produced by `test_models` within each flavour folder, eg within `RiskPaths/current/ompp-win`:

- `tombstone.txt` - Contains basic information about the compiler, members, ini file, etc. used to build and run the model. `test_models` reports if this file differs between Current and Reference.
- `logs/build.log` - A copy of captured console output from all stages of the build process.
- `logs/run.log` - A copy of the log file produced by the model when run.
- `outputs/trace.txt` - A normalized version of the event trace file, if produced by the model.
- `outputs/0_MODEL_INI.txt` - The contents of the model ini file used to run the model, if any.
- `outputs/0_SIGNIFICANT_DIGITS.txt` - Contains the number of significant digits used to round model output tables before computing digests.
- `outputs/digests.txt` - Contains the name and digest value of each file in the outputs folder (excluding subfolders).
- `outputs/*.csv` - Model output tables, rounded.
- `outputs/original/*.csv` - Model output tables, original (unrounded).

[back to output files]
[back to topic contents]

table outputs

Each table is saved in a flat csv format with a header line. Zero-based indexes identify the table cell. Each row consists of all classificatory

dimension indices, followed by the single measure dimension index, followed by the cell value. The order of classificatory dimensions is the same as the order in the `table` statement which declared the table in model code. Classificatory dimensions will include an additional trailing level for the margin, if a margin is specified for the dimension in the table declaration. The measure index is always present even if there is only one measure in the measure dimension. The cell value can be empty. Here's an example extract:

```
Dim0,Dim1,Dim2,Dim3,Value
0,0,0,0,258.453
0,0,0,1,1119.96
0,0,0,2,933.714
0,0,0,3,86.1511
0,0,1,0,172.302
...
```

[back to output files]
[back to topic contents]

event trace

=== under construction ===

Note that times >= 32767 will be shown as 99999, as will +inf, to improve comparability between ompp and Modgen event trace files.

Note options which control normalized event trace operation.

Ompp models must set EventTrace.Style to modgen for event trace files to be normalized and comparable.

[back to output files]
[back to topic contents]

Example 1 Preparing the default run

This example illustrates how to arrange that the default `test_models` run is fast, and is easier to debug or investigate if run-time anomalies or unexpected outputs occur during model development. The underlying idea is to develop a model in small incremental steps, and use `test_models` to "fail fast, fail early", after each incremental change to the model. This approach is efficient only if the testing phase is fast. This example uses the `OncoSimX` model on the Windows platform.

Step 1 investigates how the model performs with a normal Default run. First, a fresh version of the model is created using git to clone from the git server to the temporary folder `C:/Temp/OncoSimX`. Next, `test_models` is run with the `--timing` option to evaluate the performance of the Default run.

```
C:\temp\OncoSimX>%OM_ROOT%\bin\test_models -m .. --nomodgen --nocomp --timing OncoSimX
        ========================
         test_models 2.1
        ========================

        Testing: OncoSimX
        ompp-win settings: compiler=omc.exe (2021-06-29 06:31 GMT) platform=x64 configuration=release

        OncoSimX: ompp-win: Build and publish model and Default scenario
        OncoSimX: ompp-win: Build time 1m 47s
 warning => OncoSimX: ompp-win: 1794 build warnings - see OncoSimX/test_models/current/ompp-win/logs/build.log
        OncoSimX: ompp-win: Run model using OncoSimX.ini
        OncoSimX: ompp-win: Run time 25m 3s
        OncoSimX: ompp-win: Convert outputs (6 digits of precision)
        OncoSimX: ompp-win: Convert time 1m 33s
        OncoSimX: ompp-win: Create digests of current outputs
        OncoSimX: ompp-win: Digest time 0m 0s
        OncoSimX: ompp-win: No Reference outputs - create using Current outputs
        OncoSimX: ompp-win: Current vs. Reference: 215 the same (of 215)
        OncoSimX: ompp-win: Compare time 0m 0s
```

The `test_models` report shows that the run itself took 25m, and that the run used the model ini file `OncoSimX.ini`. An ini file with the same name as the model is used by default if it exists. Here's the contents of that file:

```
[OpenM]
SubValues = 12
Threads = 6

[Parameter]
SimulationCases = 1000000
SimulationSeed = 16807

; Complete example of ini-file located at: props/model/ompp/Model-example.ini
```

The run had 1,000,000 cases divided among 12 members (aka replicates or sub-samples), and was accelerated by running 6 threads in parallel (the workstation for the run had 8 processor cores). As a test, 1,000,000 cases does explore many conditions in the model and can produce statistically meaningful results, but a run of that size with `OncoSimX` is not suitable for quick iteration during model development. Moreover, a multi-threaded run is much more difficult to debug than a single-threaded run, making it a poor choice for testing during development. Ignoring run time, `test_models` took about 3m 15s. If the run were modified to take 1m 45s of run time instead, with one thread of execution, the total time would be about 5m, which would be acceptable for rapid testing.

Step 2 creates a new model ini file named `test_models.ini` based on the original model ini file `OncoSimX.ini` with several modifications. The number of members is changed to 1, the number of threads to 1, and the the number of cases changed somewhat arbitrarily to 5,000 to see how fast that will turn out:

```
[OpenM]
SubValues = 1
Threads = 1

[Parameter]
SimulationSeed = 1
SimulationCases = 5000
```

The choice of file name for the modified model ini file was not arbitrary. `test_models` will use by default a model ini file named `test_models.ini` if it exists.

Step 3 runs `test_models` with the new model ini `test_models.ini` which it will use by default. The `--newref` option is used to replace the previous results to avoid reporting meaningless differences.

```
C:\temp\OncoSimX>%OM_ROOT%\bin\test_models -m .. --nomodgen --nocomp --timing --newref OncoSimX
        ========================
         test_models 2.1
        ========================

        Testing: OncoSimX
        ompp-win settings: compiler=omc.exe (2021-06-29 06:31 GMT) platform=x64 configuration=release

        OncoSimX: ompp-win: Deleting previous Reference information
        OncoSimX: ompp-win: Build and publish model and Default scenario
        OncoSimX: ompp-win: Build time 1m 46s
 warning => OncoSimX: ompp-win: 1794 build warnings - see OncoSimX/test_models/current/ompp-win/logs/build.log
        OncoSimX: ompp-win: Run model using test_models.ini
        OncoSimX: ompp-win: Run time 1m 41s
        OncoSimX: ompp-win: Convert outputs (6 digits of precision)
        OncoSimX: ompp-win: Convert time 1m 19s
        OncoSimX: ompp-win: Create digests of current outputs
        OncoSimX: ompp-win: Digest time 0m 1s
        OncoSimX: ompp-win: No Reference outputs - create using Current outputs
        OncoSimX: ompp-win: Current vs. Reference: 215 the same (of 215)
        OncoSimX: ompp-win: Compare time 0m 0s
```

The report shows that `test_models.ini` was used as intended. The total time for `test_models` to build, run, and compare results was 4m 47s, which is about what was targeted as a quick test for incremental model development.

Step 4 uses git to add `test_models.ini` to the local repository in `C:/Temp/OncoSimX`, and then push the change to the server repository to persist it and make it available to others on the modelling team.

Step 5 runs `test_models` a final time using `test_models.ini` to see how long the Modgen version takes.

```
C:\Development\X\ompp\Perl>perl test_models.pl -m ../../models --noompp --timing OncoSimX
        ========================
         test_models 2.1
        ========================

        Testing: OncoSimX
        modgen settings: version=12,1,3,0 (2019-12-19 20:31 GMT) platform=Win32 configuration=release

        OncoSimX: modgen: Build model and prepare Default scenario
        OncoSimX: modgen: Build time 0m 15s
 warning => OncoSimX: modgen: 1 build warnings - see OncoSimX/test_models/current/modgen/logs/build.log
        OncoSimX: modgen: Run model using test_models.ini
        OncoSimX: modgen: Run time 38m 54s
 warning => OncoSimX: modgen: 1 run warnings - see OncoSimX/test_models/current/modgen/logs/run.log
        OncoSimX: modgen: Convert outputs (6 digits of precision)
        OncoSimX: modgen: Convert time 12m 3s
        OncoSimX: modgen: Create digests of current outputs
        OncoSimX: modgen: Digest time 0m 1s
        OncoSimX: modgen: Current vs. Reference: 205 the same (of 205)
        OncoSimX: modgen: Compare time 0m 0s
```

Even with the small run in `test_models.ini`, the Modgen version takes 50m 13s, making it not suitable for quick development tests for the `OncoSimmX` model with all tables. The Modgen version simulates quickly, but writing all table outputs and converting those outputs takes a long time. The reason is that Modgen uses a Microsoft Access database for storing tables, and MS Access is slow.

The folder `C:/Temp/OncoSimX` is now deleted, having served its purpose.

[back to topic contents]

Example 2 A new OpenM++ release

In this example, a new OpenM++ release has been announced, and a model developer is asked to verify that the team's model successfully builds and runs with the new version and produces identical results. The model developer uses a Windows workstation.

Step 1 downloads and extracts the new version of OpenM++, released on 2021-06-29, to a new folder on the workstation. To simplify changing between OpenM++ versions, the model developer previously created a folder named `C:/ompp_versions` on the workstation. A new command prompt is opened and used to verify the contents of that folder after the download and extraction:

```
C:\ompp_versions>dir
 Volume in drive C is OS
 Volume Serial Number is 14E2-D15F

 Directory of C:\ompp_versions

2021-06-29  07:27 AM

                .
        2021-06-29  07:27 AM

                        ..
                2021-06-12  09:49 PM

                        openmpp_win_20210505
                2021-06-12  09:50 PM

                        openmpp_win_20210602
                2021-06-12  09:50 PM

                        openmpp_win_20210612
                2021-06-29  07:27 AM

                        openmpp_win_20210629
                    0 File(s)              0 bytes
                    6 Dir(s)  1,717,655,265,280 bytes free
```

A command is issued to verify the version of OpenM++ currently in use on the workstation:

```
C:\ompp_versions>echo %OM_ROOT%
C:\ompp_versions\openmpp_win_20210602
```

The version currently in use is 2021-06-02 which is two versions old because the development team decided to skip the mid-month OpenM++ bug-fix release because that bug was known not to affect the team's model.

Step 2 prepares the test. The `OncoSimX` model is used in this example and it's assumed that the model developer is working actively on the model and has unchecked code changes. To not interfere with that current work-in-progress, git is used to create a new temporary clone of the

stable trunk version of `OncoSimX` in the temporary folder `C:/Temp/OncoSimX`.

Step 3 invokes `test_models` to create a Reference run using the team's working version (2021-06-02) of OpenM++:

```
C:\ompp_versions>cd C:\Temp\OncoSimX
C:\temp\OncoSimX>perl %OM_ROOT%/perl/test_models.pl -m .. --nomodgen --nocomp OncoSimX
        ========================
         test_models 2.0
        ========================

        Testing: OncoSimX
        ompp-win settings: compiler=omc.exe (2021-06-02 03:23 GMT) platform=x64 configuration=release

        OncoSimX: ompp-win: Build and publish model and Default scenario
 warning => OncoSimX: ompp-win: 1794 build warning(s) - see OncoSimX/test_models/current/ompp-win/logs/build.log
        OncoSimX: ompp-win: Run model using test_models.ini
        OncoSimX: ompp-win: Convert outputs (7 digits of precision)
        OncoSimX: ompp-win: Create digests of current outputs
        OncoSimX: ompp-win: No Reference outputs - create using Current outputs
        OncoSimX: ompp-win: Current vs. Reference: 215 the same (of 215)
```

The `-m ..` option specifies that the folder containing models is the parent folder of the current folder.

Step 4 switches to the new OpenM++ release dated 2021-06-29 by changing the `OM_ROOT` environment variable:

```
C:\temp\OncoSimX>set OM_ROOT=C:\ompp_versions\openmpp_win_20210629
```

and invokes `test_models` again using the same arguments:

```
C:\temp\OncoSimX>perl %OM_ROOT%/perl/test_models.pl -m .. --nomodgen --nocomp OncoSimX
        ========================
         test_models 2.1
        ========================

        Testing: OncoSimX
        ompp-win settings: compiler=omc.exe (2021-06-29 06:31 GMT) platform=x64 configuration=release

        OncoSimX: ompp-win: Build and publish model and Default scenario
 warning => OncoSimX: ompp-win: 1794 build warnings - see OncoSimX/test_models/current/ompp-win/logs/build.log
        OncoSimX: ompp-win: Run model using test_models.ini
        OncoSimX: ompp-win: Convert outputs (6 digits of precision)
        OncoSimX: ompp-win: Create digests of current outputs
        OncoSimX: ompp-win: Current vs. Reference: tombstone info differs:
        OncoSimX: ompp-win:   Reference: compiler=omc.exe (2021-06-02 03:23 GMT) platform=x64 configuration=release members=1
        OncoSimX: ompp-win:   Current:   compiler=omc.exe (2021-06-29 06:31 GMT) platform=x64 configuration=release members=1
        OncoSimX: ompp-win: Current vs. Reference: 214 the same (of 215)
DIFFERS ===> OncoSimX: ompp-win: Current vs. Reference:   1 differ
DIFFERS ===> OncoSimX: ompp-win: Current vs. Reference: DIFFERS:        0_SIGNIFICANT_DIGITS.txt
```

`test_models` reports that the OpenM++ compiler versions for the Reference and Current runs differ, as expected. Unexpectedly, it reported a single differing file. The name of that file and the two `test_models` reports pinpoint the cause: The default number of digits used by `test_models` was changed from 7 digits to 6 digits between the two OpenM++ releases. The change in the number of significant digits turned out not to affect the rounded versions of the model output tables in the two runs.

The test, at this point, has established that the model builds and runs successfully with the new version of OpenM++ and produces identical outputs for the small fast run used normally by `test_models` during `OncoSimX` development.

In step 5, the model developer decides to perform a more demanding verification by comparing results between the two OpenM++ releases using a larger run of 1,000,000 cases and 12 replicates. These are the settings in the model ini file `OncoSimX/OncoSimX.ini`:

```
[OpenM]
SubValues = 12
Threads = 6

[Parameter]
SimulationCases = 1000000
SimulationSeed = 16807

; Complete example of ini-file located at: props/model/ompp/Model-example.ini
```

The OpenM++ version is switched back to the older 2021-06-02 version

```
C:\temp\OncoSimX>set OM_ROOT=C:\ompp_versions\openmpp_win_20210602
```

and `test_models` is invoked, this time explicitly specifying the number of significant digits as well as the model ini file to use. The `--newref` option is used to create a new Reference run using the larger run specified in the ini file.

```
C:\temp\OncoSimX>perl %OM_ROOT%/perl/test_models.pl -m .. --nomodgen --nocomp --significant_digits 6 --ini OncoSimX.ini --newref OncoSimX
          ========================
           test_models 2.0
          ========================

          Testing: OncoSimX
          ompp-win settings: compiler=omc.exe (2021-06-02 03:23 GMT) platform=x64 configuration=release

          OncoSimX: ompp-win: Deleting previous Reference information
          OncoSimX: ompp-win: Build and publish model and Default scenario
 warning => OncoSimX: ompp-win: 1794 build warning(s) - see OncoSimX/test_models/current/ompp-win/logs/build.log
          OncoSimX: ompp-win: Run model using OncoSimX.ini
          OncoSimX: ompp-win: Convert outputs (6 digits of precision)
          OncoSimX: ompp-win: Create digests of current outputs
          OncoSimX: ompp-win: No Reference outputs - create using Current outputs
          OncoSimX: ompp-win: Current vs. Reference: 215 the same (of 215)
```

The OpenM++ version is switched once again to the new release 2021-06-29

```
C:\temp\OncoSimX>set OM_ROOT=C:\ompp_versions\openmpp_win_20210629
```

and `test_models` is run again to produce a comparable Current run and compare it to the Reference run:

```
C:\temp\OncoSimX>perl %OM_ROOT%/perl/test_models.pl -m .. --nomodgen --nocomp --significant_digits 6 --ini OncoSimX.ini OncoSimX
          ========================
           test_models 2.1
          ========================

          Testing: OncoSimX
          ompp-win settings: compiler=omc.exe (2021-06-29 06:31 GMT) platform=x64 configuration=release

          OncoSimX: ompp-win: Build and publish model and Default scenario
 warning => OncoSimX: ompp-win: 1794 build warnings - see OncoSimX/test_models/current/ompp-win/logs/build.log
          OncoSimX: ompp-win: Run model using OncoSimX.ini
          OncoSimX: ompp-win: Convert outputs (6 digits of precision)
          OncoSimX: ompp-win: Create digests of current outputs
          OncoSimX: ompp-win: Current vs. Reference: tombstone info differs:
          OncoSimX: ompp-win:   Reference: compiler=omc.exe (2021-06-02 03:23 GMT) platform=x64 configuration=release members=12
          OncoSimX: ompp-win:   Current:   compiler=omc.exe (2021-06-29 06:31 GMT) platform=x64 configuration=release members=12
          OncoSimX: ompp-win: Current vs. Reference: 215 the same (of 215)
```

The larger run executed successfully with the new OpenM++ release, and all output results were the same to 6 significant digits.

In this example, the `OM_ROOT` environment variable was changed using the `set` command (not the `setx` command) to select an OpenM++ version. The changed value of `OM_ROOT` applied only to commands issued in the command prompt window used for the test. The value of `OM_ROOT` in other command prompt windows or Visual Studio sessions was unaffected by the test.

[back to topic contents]

Example 3 A model code change

=== under construction ===

[back to topic contents]

Example 4 When results differ

=== under construction ===

[back to topic contents]

Example 5 Exercising a model in Debug

In this example, `test_models` is used to build and run Debug versions of a model, using both OpenM++ and Modgen versions. Debug versions run more slowly but contain more run-time checks which can identify errors, so it is a good idea to test the Debug versions from time to time. `test_models1 makes that easy. This example uses the` OncoSimX` model running on a Windows system.

In step 1, a fresh set of Reference runs are created:

The Modgen run took considerably longer than the OpenM++ run.

[back to topic contents]

Example 6 A performance comparison

In this example, `test_models` is used to evaluate the performance difference between two versions of the same model, holding other effects constant. Specifically, this example seeks to measure the performance cost of the run-time checks which OpenM++ performs to ensure that attributes are not modified directly or indirectly in event time functions. Although those runtime checks are core OpenM++ functionality, the OpenM++ option `verify_attribute_modification` can be used to remove them.

Step 1 sets up the experiment. The `OncoSimX` model is chosen because it is a large-scale production model with many events. The experiment is done on a secondary (older) Windows workstation which will run no other tasks during the experiment. The experiment is temporary, so a single git command is issued to create a new clone of `OncoSimX` in a convenient temporary location, which in this example is `C:/Temp/OncoSimX`. The `OncoSimX` clone contains no `test_models` results, which is appropriate for a model git repository. It does contain the model ini file `test_models.ini` in the model root.

Step 2 sets up the `test_models` model run to serve as a performance test. Normally, a model run for `test_models` is small and fast. This experiment requires a run which is large enough to produce reliable performance information. The `OncoSimX` model ini file `test_models.ini` produces small fast runs for rapid development. The contents are:

```
[OpenM]
SubValues = 1
Threads = 1

[Parameter]
SimulationSeed = 1
SimulationCases = 5000

[Tables]
;Retain = \
;     TG01_Breast_Cancer_Tables
```

A single line is modified to increase the number of cases from 5,000 to 1,000,000 for the experiment.

```
SimulationCases = 1000000
```

Step 3 uses `test_models` to create a Reference run to serve as a basis for comparison. A command window is opened and `test_models` is invoked:

```
perl test_models.pl -m C:/Temp --timing --nomodgen OncoSimX
```

The temporary `OncoSimX` clone was created in the folder `C:/Temp`, so that folder is specified as the models folder using the `-m` option. Timing information is requested using the `--timing` option. The experiment is being done on a Windows workstation with Modgen installed, so the `--nomodgen` argument is used to suppress that flavour.

The output is

```
        ========================
         test_models 2.1
        ========================

        Testing: OncoSimX
        ompp-win settings: compiler=omc.exe (2021-07-01 12:27 GMT) platform=x64 configuration=release

        OncoSimX: ompp-win: Build and publish model and Default scenario
        OncoSimX: ompp-win: Build time 3m 58s
warning => OncoSimX: ompp-win: 1794 build warnings - see OncoSimX/test_models/current/ompp-win/logs/build.log
        OncoSimX: ompp-win: Run model using test_models.ini
        OncoSimX: ompp-win: Run time 98m 8s
        OncoSimX: ompp-win: Convert outputs (6 digits of precision)
        OncoSimX: ompp-win: Convert time 2m 10s
        OncoSimX: ompp-win: Create digests of current outputs
        OncoSimX: ompp-win: Digest time 0m 1s
        OncoSimX: ompp-win: No Reference outputs - create using Current outputs
        OncoSimX: ompp-win: Current vs. Reference: 215 the same (of 215)
        OncoSimX: ompp-win: Compare time 0m 0s
        OncoSimX: Flavour compare time 0m 0s
```

Note that `test_models` created Reference results from Current results because no Reference run was present. The reported run time was 98m 8s.

Step 4 modifies the model to remove the normal run-time checks for the experiment. Because this is a temporary clone of OncoSimX which will be discarded after the experiment, the source code can be modified freely with no consequences. The small source file `TraceOptions.mpp` was selected arbitrarily to implement the experiment, and the following line added:

```
options verify_attribute_modification = off;
```

Step 5 uses `test_models` to create a Current run which will be compared to the Reference run. The invocation of `test_models` is identical to the previous invocation.

```
perl test_models.pl -m C:/Temp --timing --nomodgen OncoSimX
```

The output is

```
        ========================
         test_models 2.1
        ========================

        Testing: OncoSimX
        ompp-win settings: compiler=omc.exe (2021-07-01 12:27 GMT) platform=x64 configuration=release

        OncoSimX: ompp-win: Build and publish model and Default scenario
        OncoSimX: ompp-win: Build time 3m 58s
warning => OncoSimX: ompp-win: 1794 build warnings - see OncoSimX/test_models/current/ompp-win/logs/build.log
        OncoSimX: ompp-win: Run model using test_models.ini
        OncoSimX: ompp-win: Run time 95m 16s
warning => OncoSimX: ompp-win: 1 run warnings - see OncoSimX/test_models/current/ompp-win/logs/run.log
        OncoSimX: ompp-win: Convert outputs (6 digits of precision)
        OncoSimX: ompp-win: Convert time 2m 12s
        OncoSimX: ompp-win: Create digests of current outputs
        OncoSimX: ompp-win: Digest time 0m 0s
        OncoSimX: ompp-win: Current vs. Reference: 215 the same (of 215)
        OncoSimX: ompp-win: Compare time 0m 0s
        OncoSimX: Flavour compare time 0m 0s
```

All model outputs are identical as expected. `test_models` reports one run-time warning which was absent in the first run. That's because OpenM++, by design, emits a warning on every run of a model which disables run-time checking of event times. `test_models` noted the warning and reported it when it analyzed the run log. The reported run time was 95m 16s.

Step 6 compares performance between the Reference and Current runs. The model run time in the Current run was 172s less than the Reference run, and shows that about 2.9% of the elapsed time of the Reference run was overhead associated with the run-time checks. That is a realistic assessment of the overhead of the run-time checks for `OncoSimX` when used in production.

The elapsed time reported by `test_models` includes the time required to read parameters and to assemble and write tables, which can be significant. It might also be interesting to assess the performance effect on the simulation time alone, removing the fixed cost of parameter reading and table assembly and writing. The model log files saved by `test_models` can be used to perform that calculation.

Here's a line extracted from the Reference log file, located at `C:/Temp/OncoSimX/test_models/reference/ompp-win/logs/run.log`:

```
2021-07-01 10:16:52.928 member=0 Simulation summary: cases=1000000, events/case=1128.9, entities/case=5.4, elapsed=5693.747633s
```

and here's the corresponding line extracted from the Current log file, located at `C:/Temp/OncoSimX/test_models/current/ompp-win/logs/run.log`.

```
2021-07-01 12:31:45.777 member=0 Simulation summary: cases=1000000, events/case=1128.9, entities/case=5.4, elapsed=5525.430919s
```

Conveniently, this line in the OpenM++ model log reports the elapsed time of the computational phase, in seconds. The simulation time was 5694s for the Reference run and 5525s for the Current run, for a difference of 168s. So about 3.0% of the computation time was overhead associated with the run-time checks. That's only slightly higher than the proportion computed using elapsed time because the time required to read parameters and to assemble and write tables (about 190s) is a relatively small portion of the elapsed time of an `OncoSimX` run of 1,000,000 (5525s).

[back to topic contents]

Example 7 Using event trace output

=== under construction ===

The following applies to a case-based model.

If test_models detects unexpected output differences which are not immediately explained by recent model code changes or by perusing differences in specific tables, it can be useful to determine whether the differences are due to a handful of different cases in the run, or to some other cause.

To find out, turn on the case checksum option by adding the statement

options case_checksum = on;

to model code. In the RiskPaths example model, this is done in a dedicated model source file named `TraceOptions.mpp` whcih is normally:

```
//LABEL(TraceOptions, EN) Control options for trace output

/* NOTE(TraceOptions, EN)
   The options must be set to off for normal use.
   If the options are on, very large trace output files may be produced by the model.
*/

options event_trace = off;
options case_checksum = off;
```

Turning `case_checksum` on will cause a trace file to be generated when test_models runs the model which contains one line for each case. For RiskPaths, the first lines of the trace file look like this:

```
Seed:            1 Sample:  0 Checksum: 4.4094661700e+02
Seed:    470583131 Sample:  0 Checksum: 4.1293225500e+02
Seed:   1715377866 Sample:  0 Checksum: 3.4860055300e+02
Seed:    430166727 Sample:  0 Checksum: 1.3482143400e+02
Seed:    743781969 Sample:  0 Checksum: 1.4831547800e+02
Seed:    594673803 Sample:  0 Checksum: 4.1803896100e+02
Seed:   2048386738 Sample:  0 Checksum: 3.5386231900e+02
Seed:   1681919254 Sample:  0 Checksum: 4.0119902000e+02
Seed:     78320563 Sample:  0 Checksum: 3.9439706500e+02
Seed:   1637697257 Sample:  0 Checksum: 4.9065368400e+02
Seed:    127732046 Sample:  0 Checksum: 9.8715092900e+02
Seed:   2114816392 Sample:  0 Checksum: 3.9576701300e+02
Seed:   1616242686 Sample:  0 Checksum: 1.2478963010e+03
Seed:   1222943222 Sample:  0 Checksum: 4.5861339400e+02
Seed:    170881636 Sample:  0 Checksum: 1.4320257000e+02
```

This file is noticed by test_models and will participate in test_models comparisons. Test_models saves the trace file under the name `trace.txt` in the same folder where it places csv versions of tables.

If `test_models` reports that the file `trace.txt` differs between the ompp and modgen versions, then one or more cases differ in the two flavours. Use a tool or text editor to compare the two versions of `trace.txt` to determine the seed of the first case which has a differing checksum in the two variants.

More to follow...

[back to topic contents]

Time-like and Event-like Attributes

This topic describes the distinction between event-like and time-like attributes, and forbidden uses of time-like attributes in model code.

Related topics

- Model Code

Topic contents

- Time-like and event-like attributes: Definitions and examples
- Potential issues: How time-like attributes could violate the model specification
- Protection against potential issues: How OpenM++ protects against these issues
- Disabling protection: How to disable OpenM++ run-time protection

Time-like and event-like attributes

An OpenM++ simulation consists of a sequence of events which change values of attributes when the events occur. Attributes are observed only at events, not between events. Some attributes, however, can undergo unobserved changes between events. An example is the built-in attribute `time`, which changes continuously between events, even though it is observed only when events occur. Attributes like `time` or `age` which undergo unobserved changes between events are called *time-like*.

Another example of a time-like attribute is the identity attribute

```
bool first6months = (age <= 0.5);
```

Logically, the attribute `first6months` changes value when `age` is exactly equal to 0.5. However, unless there is an event which occurs precisely at that moment, the transition of `first6months` from `true` to `false` will not be observed when it logically occurs. For example, if the first event in the entity occurs at age 1.0, the value of `first6months` will become `false` at that time, but the transition of `first6months` from `true` to `false` at age 0.5 will not have been noted at the time it logically occurred.

An attribute which changes only at events is called *event-like*. For example, the attribute

```
bool is_alive = {true);
```

is event-like. It has initial value `true` and changes to `false` when the `Mortality` event occurs (code not shown).

All self-scheduling attributes are event-like because they schedule an internal event at appropriate times. For example, `self_scheduling_int(age)` is an event-like attribute with value equal to integer age of the entity. It is event-like because it schedules an internal event when `age` will attain its next integer value.

A derived attribute which observes a time-like attribute may be event-like. For example, the derived attribute

```
value_at_first_entrance(is_alive, false, age)
```

which records age at death is event-like because it observes the value of `age` when `is_alive` transitions to `false` during the Mortality event.

The OpenM++ compiler deduces whether an attribute is event-like or time-like. The Symbol Reference of a model can be consulted to determine if an attribute is time-like.
For example, the Symbol Reference topic for the `age` attribute in `RiskPaths` looks like

and indicates, after **Kind:**, that it is time-like. Attributes not indicated as time-like in the Symbol Reference are event-like.

[back to topic contents]

Potential issues

There are three situations where use of a time-like attribute could violate the model specification and produce unexpected or incorrect model results.

1. A time-like filter of an entity table or entity set;
2. A time-like dimension of an entity table or entity set;
3. The use of a time-like attribute in an event time function.

For situation 1, consider the following table declaration using the time-like attribute `first6months` introduced above.

```
table Host MyTable
[ first6months ]
{
   {
      duration() //EN duration
   }
};
```

Logically, this table should sum the value 0.5 for each entity which survives to age 0.5 or beyond. But if the first event occurs later, for example when `age` is 1.0, the value 1.0 would be summed instead, producing a logically erroneous result given the table specification.

For situation 2, consider a modified version of this table which uses the time-like attribute `first6months` as a dimension instead of as a filter. The total duration in the table would be correct, but it would be misclassified between the two cells in the table.

For situation 3, consider if the time-like attribute `first6months` is used in an event time function. Assume that the model implements a first tranche of vaccination when an infant is 6 months old. Model code in the time function of the `ChildVaccination` event might include a statement like

```
if (!vaccinated && !first6months) return WAIT(0);
```

to trigger the `ChildVaccination` event when the child is 6 months old. However, the attribute `first6months` is time-like and changes value at 6 months of age only if some other event happens to occur at that precise value of `age`. If there is no such event, the `ChildVaccination` event would be triggered at some later time when the value of `first6months` is next updated, perhaps at the first birthday when `age` is 1.0.

[back to topic contents]

Protection against potential issues

OpenM++ identifies situations 1 and 2 at build time and issues an error message like

```
error : the filter of table 'TimeLikeTest' must not be time-like
error : dimension 'first6months' of table 'TimeLikeTest' must not be time-like
```

OpenM++ identifies situation 3 at run time and halts the simulation with an error message such as the following:

```
Simulation error: Attempt to access the time-like attribute age
 by the event time function of event OneTime
 in entity_id 2
 when current time is -inf
 before enter_simulation
 in simulation member 0
 with combined seed 1
```

Modgen-specific: In Modgen parlance a time-like attribute is called a continuously-updated state (CUS). Modgen issues an error message at build time for situations 1 and 2. Modgen attempts to detect situation 3 at build time, by prohibiting some types of C++ code (including links/pointers) in event time functions, and by scanning event time functions for any symbol which has the same name as an attribute. The OpenM++ design takes a different approach, instead imposing no restrictions on model C++ code and detecting situation 3 directly at run-time.

[back to topic contents]

Disabling protection

There is some overhead associated with run-time detection of situation 3 which is typically minor. It can be eliminated by disabling run-time detection using the statement

```
options verify_timelike_attribute_access = off;
```

If this run-time detection is disabled, the model will write the following warning to the log on every model run:

```
Warning : prohibited time-like attribute access is not detected with verify_timelike_attribute_access = off
```

[back to topic contents]

Use Modules

Topic summary, two sentences max.

Related topics

- Model Code

Topic contents

- Subtopic #1 Short description of Subtopic 1

Subtopic 1

Subtopic 1 content.

[back to topic contents]

Weighted Tabulation

For case-based models, the `weighted_tabulation` option creates, for each entity, the built-in attribute `entity_weight` which scales the entity's contribution to tables.

Related topics

- Model Code
- Population Size and Scaling

Topic contents

- Introduction and Background
- Syntax and Use How to activate and use
- Limitations Limitations of the current implementation
- *Modgen-specific* Modgen issues Weighted tabulation in a x-compatible model

Introduction and Background

Some case-based microsimulation models use micro-data directly from a survey, census, or administrative source. Micro-data from such sources often has a weight associated with each observation which reflects the sampling design and maybe post-stratification or under-count adjustment. Case weights can also be useful in microsimulation models which are not based on micro-data. Such models instead generate cases synthetically from multivariate distributions. They may deliberately over-sample portions of the synthetic population of particular interest, and then adjust for that oversampling by assigning a case weight equal to the reciprocal of the oversampling factor.

OpenM++ contains optional functionality to associate a weight with each entity. That weight scales the contribution of the entity to table counts and sums. The functionality facilitates assigning the same case weight to all of the entities in a case for table coherence. This is important for models which have multiple entities in each case, e.g. ancillary family members of a core entity which may be created later in the simulation of the case. The design integrates with population scaling by computing and using the sum of case weights.

A time-based microsimulation model simulates interacting entities. It is unclear how one might validly represent an interaction of entities which have non-equal weights. Instead, for time-based models based on weighted micro-data, a micro-data record is typically cloned or sampled based on its weight to produce a starting population of entities whose weights are all equal. Such an equal-weighted population can represent a real-world population of a different size by using population scaling, rather than by assigning a weight to each entity with all weights being equal. The end result is the same, but population scaling is more efficient for memory and computation compared to identical entity weights. Also, it is not clear how to implement population scaling in a time-based model with entity weights if the model contains entities of different types, e.g. a single special `Ticker` entity, or multiple `Dwelling`, `Person`, and `Family` entities, or a fixed number of `Region` entities. For these reasons, entity weights are forbidden in time-based models in OpenM++. Use population scaling to make a time-based model represent a real population of a different size. See Population Size and Scaling for more information.

[back to topic contents]

Syntax and Use

By default, entities are unweighted. To activate entity weights, include the statement

```
options weighted_tabulation = on;
```

in the source code of a case-based model. A natural place to insert this statement is the module `ompp_framework.ompp`. If weighting is turned on in a time-based model, an error message like the following is emitted:

```
error : weighted tabulation is not allowed with a time-based model, use population scaling instead.
```

There may be situations where it makes sense for a time-based model to have entity weights. For example, a model with a non-interacting population of weighted entities might be time-based to implement dynamic calibration to exogenous control totals. To override the default behaviour and allow weighted tabulation for a time-based model, add the following statement (as well as the above statement):

```
options weighted_tabulation_allow_time_based = on;
```

When weighting is turned on, each entity has a new built-in attribute named `entity_weight`, of type `double`. Usually model code does not assign a value directly to `entity_weight`. Instead, before entities are created for a case, model code sets the initial value of `entity_weight` for all entities in the case by calling the function `set_initial_weight`, as in the following contrived example:

```
void CaseSimulation(case_info &ci)
{
   extern void SimulateEvents(); // defined in a simulation framework module

   // Provide the weight used to initialize the entity_weight attribute for new entities
   set_initial_weight(2.0);

   // For Modgen-compatible models, use the following instead
   //SetCaseWeight(2.0);

   // Initialize the person entity
   auto prPerson = new Person();
   prPerson->Start();

   // Simulate events until there are no more.
   SimulateEvents();
}
```

Calling `set_initial_weight` before creating any entities in the case ensures that the built-in attribute `entity_weight` will have that same value for all entities in the case. The call to `set_initial_weight` also enables the calculation of the sum of case weights. That sum of weights is used to correctly scale the population to a specified size if the model uses both weights and population scaling. For that to work correctly, `set_initial_weight` must be called once and only once in the logic of the case, before any entities in the case are created.

If weighted tabulation is not enabled, entities have no attribute named `entity_weight`, and calls to `set_initial_weight` have no effect (but are benign).

If weighted tabulation is enabled, but `set_initial_weight` is not called before creating entities in the case, the `entity_weight` attribute will be 1.0. However, the total sum of weights used for population scaling will be incorrect because the calculation depends internally on the call to `set_initial_weight`. Ensure that model code calls `set_initial_weight` once and only once before creating entities in the case.

[back to topic contents]

Limitations

Weighted tabulation works for table statistics based on counts and sums. It does not work yet for ordinal statistics such as the median or the gini coefficient. Such statistics will be computed ignoring weights, i.e. as though all weights are 1.0. If a table uses an ordinal statistic and `weighted_tabulation` is on, the OpenM++ compiler will issue a warning. For example, the table

```
table Person DurationOfLife //EN Duration of Life
{
   {
      value_in(alive),              //EN Population size
      min_value_out(duration()),     //EN Minimum duration of life decimals=4
      max_value_out(duration()),     //EN Maximum duration of life decimals=4
      duration() / value_in(alive),  //EN Life expectancy decimals=4
      P50(value_out(duration()))     //EN Median duration of life decimals=4
   }   //EN Demographic characteristics
};
```

would emit an error like

```
error : weighting is not supported for statistic 'P50' in table 'DurationOfLife', consider using untransformed ...
```

[back to topic contents]

Modgen issues

case-based models (Modgen)

Modgen implements similar case weighting functionality and weight-based population scaling to OpenM++ using a function named `SetCaseWeight`. X-compatible models can call `SetCaseWeight` instead of `set_initial_weight` as in the commented statement in the previous example. The OpenM++ framework supplies versions of `SetCaseWeight` which call `set_initial_weight` internally.

OpenM++ functions intrinsically at the sub-sample/replicate/member level, so the notion of a distinct total weight and sub-sample weight does not

apply in OpenM++.

time-based models (Modgen)

Modgen does not implement population scaling for time-based models. To work around this limitation, model developers have called the Modgen function `Set_actor_weight` in actor `Start` functions to scale results to represent a larger population. Consider a time-based model which includes two exogenous parameters, `StartingPopulationRealSize` for the size of the true real-world population which is represented by the model, and `StartingPopulationSize` for the size (number of entities) of the synthetic starting population in the model. The Modgen approach might look like this:

```
void Person::Start()
{
   // Initialize all attributes (OpenM++).
   initialize_attributes();

   // The following function calls implement population scaling for Modgen,
   // using identical weights for each Person entity in the simulation.
   // These calls do nothing in OpenM++.
   // OpenM++ can implement population scaling directly for time-based models.

   double dWeight = (double) StartingPopulationRealSize / (double) StartingPopulationSize;
   Set_actor_weight( dWeight );
   Set_actor_subsample_weight( dWeight );
...
```

The OpenM++ framework includes do-nothing versions of the Modgen functions `Set_actor_weight` and `Set_actor_subsample_weight` so this same code will build without error in OpenM++.

To perform the identical population scaling directly in the OpenM++ version of the model (without weights), include the following statement in `ompp_framework.ompp`:

```
use "time_based/time_based_scaling_exogenous.ompp";
```

That `use` module integrates with the OpenM++ framework to scale table counts and sums by the factor

```
(double) StartingPopulationRealSize / (double) StartingPopulationSize
```

using the exogenous parameters `StartingPopulationRealSize` and `StartingPopulationSize`.

These two parameters are already declared in the `use` module `time_based_scaling_exogenous.ompp` in OpenM++. Declare them in the Modgen version using a Modgen-only source code file name, for example `modgen_PopulationSize.mpp`, with content

```
parameters
{
   //EN Simulation population size
   int StartingPopulationSize;

   //EN True population size
   double StartingPopulationRealSize;
};
```

and then make the values of these two parameters available to both Modgen and OpenM++ by placing them in a file processed by both, for example `PopulationSize.dat` with contents like

```
parameters
{
   //EN Simulation population size
   int StartingPopulationSize = 25000;

   //EN True population size
   double StartingPopulationRealSize = 10000000;
};
```

For more about the visibility of model source code and parameter value files in OpenM++ and Modgen, see Model Code. For more about population scaling in OpenM++, see Population Size and Scaling.

[back to topic contents]

File-based Parameter Values

This topic describes how parameter values can be be represented in files in OpenM++. File-based parameter values are used by several OpenM++ components including build, run, UI, scripting, import, export.

Related topics

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

Topic contents

- Introduction
- How to use CSV or TSV files for input parameters values
  - CSV or TSV file with dimension(s) and parameter value(s)
  - CSV or TSV file with dimension(s) and multiple parameter sub-values
  - CSV or TSV files with IDs as dimension(s) items
  - CSV or TSV file with parameter values only

Introduction

In several contexts, the value(s) of a parameter can be in a file whose name is the name of the parameter followed by a suffix which indicates content and format.

For example, if a file named `AgeBaselinePreg1.csv` is present in the folder `RiskPaths/parameters/Default', it will be used by the OpenM++ compiler to provide values for that parameter when publishing the Default scenario when the model is built.

The suffix of the file indicates how the contents of the file are interpreted, as follows:

File suffix	Meaning
.csv	Comma-separated values (CSV) with header and dimension values, one value per line
.tsv	Tab-separated values (TSV) with header and dimension values, one value per line
.id.csv	CSV with header and dimension values as 0-based ID's
.id.tsv	TSV with header and dimension values as 0-based ID's
.value.csv	CSV with values only (no header or dimension values)
.value.tsv	TSV with values with (no header or dimension values)

The header line contains the column names of indices as specified in the parameter declaration in model code. The header line can optionally start with an initial column named `sub_id`. If so, the first value in each line is the 0-based index of the replicate (aka sub, subsample, member).

[back to topic contents]

How to use CSV or TSV files for input parameters values

You can use CSV or TSV files to supply input parameter values for your model. For example, if the RiskPaths model has the `SeparationDurationBaseline` parameter:

```
partition DISSOLUTION_DURATION    //EN Duration since union dissolution
{
  2, 6, 10, 15
};
..........
// Separation Duration Baseline of 2nd Formation
double    SeparationDurationBaseline[DISSOLUTION_DURATION] = {
  0.1995702, 0.1353028, 0.1099149, 0.0261186, 0.0456905
};
```

then you can supply parameter values from `RiskPaths/parameters/RiskPths.dat` or from `RiskPaths/parameters/SeparationDurationBaseline.csv`:

```
dim0,      param_value
"(-∞,2)",  0.1995702
"[2,6)",   0.1353028
"[6,10)",  0.1099149
"[10,15)", 0.0261186
"[15,∞)",  0.0456905
```

If parameter values are coming from CSV or TSV file then you can use Markdown file(s) to provide parameter value notes. For example, `RiskPaths/parameters/SeparationDurationBaseline.EN.md`:

```
This is a parameter values for Separation Duration Baseline of 2nd Formation.

It is a test sample model and data may not be accurate.
```

and `RiskPaths/parameters/SeparationDurationBaseline.FR.md`:

```
**Translation below created by Google, please provide a proper French translation before publishing the model**

Il s'agit d'une valeur de paramètre pour la ligne de base de la durée de séparation de la 2e formation.

Il s'agit d'un modèle d'échantillon de test et les données peuvent ne pas être exactes.
```

The following rules are applied to parameter CSV or TSV files:

- file name must be the same as parameter name, which is `SeparationDurationBaseline` in example above
- it can be parameter.csv with comma-separated values or parameter.tsv with tab-separated values
- if dimension item name or parameter value contains comma then it must be "quoted"
- header line of the file must contain dimension names and "param_value", for example: `dim0,otherDim,dim3,param_value`
- (optional) if parameter has multiple sub-values then header line must start with "sub_id" (see example below)
- if the scenario directory contains a .dat file with values for ParameterName and also a file named ParameterName.csv the CSV values will override the .dat values.

The following formats of CSV or TSV files are supported:

- CSV or TSV file with dimension(s) and parameter value(s)
- CSV or TSV file with dimension(s) and multiple parameter sub-values
- CSV or TSV files with IDs as dimension(s) items
- CSV or TSV file with parameter values only, without dimensions (omc only)

Subsequent sub-topics describe each in turn, with examples.

[back to topic contents]

CSV (or TSV) file with dimension(s) and parameter value(s)

Dimension items must be supplied as item "names", similar to `partition DISSOLUTION_DURATION` above. Or take a look into another example of two-dimensional parameter:

```
classification SEX //EN Sex
{
  FEMALE,
  MALE
};
range YEAR //EN Year
{
  2019,
  2021
};
............
double GenderByYearRatio[SEX][YEAR] = {
  0.1, (2)0.2, 0.5, 0.6, 0.7
};
```

GenderByYearRatio.csv file:

```
dim0,   dim1, param_value
FEMALE, 2019, 0.1
FEMALE, 2020, 0.2
FEMALE, 2021, 0.2
MALE,   2019, 0.5
MALE,   2020, 0.6
MALE,   2021, 0.7
```

- Line order in the file is not important and openM++ will sort it automatically.
- If there some line(s) are missing then parameter value will be a default for that parameter type, for example: 0.0.

[back to topic contents]

CSV (or TSV) file with dimension(s) and multiple parameter sub-values

It can be used for uncertainty probabilistic analysis, file example:

```
sub_id, dim0,   dim1, param_value
 0,     FEMALE, 2019, 0.1
 0,     FEMALE, 2020, 0.2
 0,     FEMALE, 2021, 0.2
 0,     MALE,   2019, 0.5
 0,     MALE,   2020, 0.6
 0,     MALE,   2021, 0.7
 1,     FEMALE, 2019, 1.1
 1,     FEMALE, 2020, 1.2
 1,     FEMALE, 2021, 1.2
 1,     MALE,   2019, 1.5
 1,     MALE,   2020, 1.6
 1,     MALE,   2021, 1.7
...........
```

- Header line must start with "sub_id" in order to indicate presnse of sub values in parameter.csv file.
- Sub value id's can be any integer values, for example: -1, 0, 4, 8, 12, 42. It must be integer but does not need to be positive or sequential.
- The first sub value id in a CSV file is considered to be the "default" sub value for that parameter.

[back to topic contents]

CSV or TSV files with IDs as dimension(s) items

You can use dimension item ID instead of item names, for example SeparationDurationBaseline.id.csv:

```
dim0, param_value
0,    0.1995702
1,    0.1353028
2,    0.1099149
3,    0.0261186
4,    0.0456905
```

GenderByYearRatio.id.csv:

```
dim0, dim1, param_value
0,    0,    0.1
0,    1,    0.2
0,    2,    0.2
1,    0,    0.5
1,    1,    0.6
1,    2,    0.7
```

Please notice file naming convention: `ParameterName.id.csv` or `ParameterName.id.tsv`

[back to topic contents]

CSV or TSV file with parameter values only

This format is only supported for parameters read at build time by the OpenM++ compiler (omc).

In this format the file contains only parameter value(s) without dimensions, for example `SeparationDurationBaseline.value.csv`:

```
0.1995702, 0.1353028, 0.1099149, 0.0261186, 0.0456905
```

`GenderByYearRatio.value.csv`:

```
0.1, 0.2, 0.2,
0.5,
0.6, 0.7
```

- Please note the file naming convention: `ParameterName.value.csv` or `ParameterName.value.tsv`
- Values in the file must be ordered according to the dimension order in the parameter declaration, ie the same order as in a .dat file.
- Any number of values may be given on each line.
- Each pair of values must be separated by a comma, including the last value on a line and the first value on the subsequent line.

[back to topic contents]

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.

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
OM_CFG_TYPE_MAX_LEN=366
```

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
- `OM_CFG_TYPE_MAX_LEN` : type size limit, if dimension type exceed this size then UI would not show parameter or output table with that dimension

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 oms root directory, default: models/bin
-oms.ModelLogDir:    models log directory, if relative then must be relative to oms root directory: default: models/log
-oms.ModelDocDir:    models documentation directory, if relative then must be relative to oms root directory: default: models/doc
-oms.HomeDir:        user personal home directory, if relative then must be relative to oms 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.FilesDir:       user files directory, if relative then must be relative to oms root directory, if home directory specified then it is: home/io
-oms.AllowMicrodata: if true then allow model run microdata
-oms.HtmlDi:         front-end UI directory, if relative then must be relative to oms 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 oms 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 oms 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.AdminAll        if true then allow global administrative routes: /admin-all/
-oms.NoAdmin         if true then disable local administrative routes: /admin/
-oms.NoShutdown      if true then disable shutdown route: /shutdown/

-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
# This is also a comment

; Ini file can be supplied to oms.exe as command line option "-ini" or "-OpenM.IniFile"
; "-ini" is a short form of "-OpenM.IniFile", command lines below are equal:
;
# oms.exe -ini           path/to/oms.ini
# oms.exe -OpenM.IniFile path/to/oms.ini

; "-l"   is a short form of "-oms.Listen", command lines below are equal:
;
# oms.exe -l          localhost:4040
# oms.exe -oms.Listen localhost:4040

; boolean options can be "true" or "false" or empty value
; boolean empty value is the same as "true"
; for example both command lines below are equal:
;
# oms -oms.ApiOnly
# oms -oms.ApiOnly true

[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
; ModelDocDir    = models/doc     # models documentation directory, default: models/doc, 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
; FilesDir       =                # user files directory, if relative then must be relative to oms root directory, if home directory specified then it is: home/io
; AllowMicrodata = false          # if true then allow model run microdata
; 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
; 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
; Languages      = en             # comma-separated list of supported languages
; CodePage       =                # code page to convert source file into utf-8, e.g.: windows-1252
; DoubleFormat   = %.15g          # format to convert float or double value to string, e.g. %.15g
; AdminAll       = false          # if true then allow global administrative routes: /admin-all/
; NoAdmin        = false          # if true then disable local administrative routes: /admin/
; NoShutdown     = false          # if true then disable shutdown route: /shutdown/

[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

; "-v" is a short form of "-OpenM.LogToConsole"

; 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 pid-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
```

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
        doc/  -> models documentation directory
```

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 -oms.ModelDocDir /my-models/doc2
```

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, model ini-files, downloads of model model run results or upload input scenarios:

```
oms -oms.HomeDir models/home -oms.AllowDownload
oms -oms.HomeDir models/home -oms.AllowDownload -oms.AllowUpload
```

Important: Please keep in mind, if you allow users to download or upload files on your server then it may create a security risk.

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
        doc/  -> models documentation directory
        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
             disk.ini  -> (optional) disk usage control settings to set storage quotas
    log/     -> recommended log files directory
    models/
        bin/  -> default model.exe and model.sqlite directory
        log/  -> default directory for models run log files
        doc/  -> models documentation directory
        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 to restrict storage size
        active/   -> active model run state files
        history/  -> model run history files
        past/     -> (optional) shadow copy of history folder, invisible to the end user
        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
            jobs.queue.all.paused  -> if such file exists then all jobs in all queues are 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 /api/model/:model/pack
```

GET model metadata including text (description and notes)

```
GET /api/model/:model/text
GET /api/model/:model/text/lang/:lang
GET /api/model/:model/pack/text
GET /api/model/:model/pack/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 output table calculated values from model run

```
POST /api/model/:model/run/:run/table/calc
```

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

```
POST /api/model/:model/run/:run/table/calc-id
```

Read output table values and compare model runs

```
POST /api/model/:model/run/:run/table/compare
```

Read output table values and compare model runs (enun id's)

```
POST /api/model/:model/run/:run/table/compare-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
```

Read aggregated microdata from model run

```
POST /api/model/:model/run/:run/microdata/calc
```

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

```
POST /api/model/:model/run/:run/microdata/calc-id
```

Read microdata run comparison

```
POST /api/model/:model/run/:run/microdata/compare
```

Read microdata run comparison (enum id's)

```
POST /api/model/:model/run/:run/microdata/compare-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 calculated expression(s) from model run

```
GET /api/model/:model/run/:run/table/:name/calc/:calc
GET /api/model/:model/run/:run/table/:name/calc/:calc/start/:start
GET /api/model/:model/run/:run/table/:name/calc/:calc/start/:start/count/:count
```

GET output table values and compare model runs

```
GET /api/model/:model/run/:run/table/:name/compare/:compare/variant/:variant
GET /api/model/:model/run/:run/table/:name/compare/:compare/variant/:variant/start/:start
GET /api/model/:model/run/:run/table/:name/compare/:compare/variant/:variant/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 aggregated microdata from model run

```
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/calc/:calc
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/calc/:calc/start/:start
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/calc/:calc/start/:start/count/:count
```

GET microdata run comparison

```
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/compare/:compare/variant/:variant
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/compare/:compare/variant/:variant/start/:start
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/compare/:compare/variant/:variant/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 calculated table expressions from model run

```
GET /api/model/:model/run/:run/table/:name/calc/:calc/csv
GET /api/model/:model/run/:run/table/:name/calc/:calc/csv-bom
```

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

```
GET /api/model/:model/run/:run/table/:name/calc/:calc/csv-id
GET /api/model/:model/run/:run/table/:name/calc/:calc/csv-id-bom
```

GET csv model runs comparison table expressions

```
GET /api/model/:model/run/:run/table/:name/compare/:compare/variant/:variant/csv
GET /api/model/:model/run/:run/table/:name/compare/:compare/variant/:variant/csv-bom
```

GET csv model runs comparison table expressions (enum id's)

```
GET /api/model/:model/run/:run/table/:name/compare/:compare/variant/:variant/csv-id
GET /api/model/:model/run/:run/table/:name/compare/:compare/variant/:variant/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 csv aggregated microdata from model run

```
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/calc/:calc/csv
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/calc/:calc/csv-bom
```

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

```
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/calc/:calc/csv-id
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/calc/:calc/csv-id-bom
```

GET csv microdata run comparison

```
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/compare/:compare/variant/:variant/csv
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/compare/:compare/variant/:variant/csv-bom
```

GET csv microdata run comparison (enum id's)

```
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/compare/:compare/variant/:variant/csv-id
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/compare/:compare/variant/:variant/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
```

POST delete multiple worksets

```
POST /api/model/:model/delete-worksets
```

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
```

POST delete model runs

```
POST /api/model/:model/delete-runs
```

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 /api/download/start/delete/:folder
```

DELETE all download files

```
DELETE /api/download/delete-all
DELETE /api/download/start/delete-all
```

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 /api/upload/start/delete/:folder
```

DELETE all upload files

```
DELETE /api/upload/delete-all
DELETE /api/upload/start/delete-all
```

Download and upload user files

GET user files tree

```
GET /api/files/file-tree/:ext/path/:path
GET /api/files/file-tree/:ext/path/
GET /api/files/file-tree/:ext/path
GET /api/files/file-tree/:ext/path?path=....
```

POST upload to user files

```
POST /api/files/file/:path
POST /api/files/file?path=....
```

PUT create user files folder

```
PUT /api/files/folder/:path
PUT /api/files/folder?path=...
```

DELETE file or folder from user files

```
DELETE /api/files/delete/:path
DELETE /api/files/delete?path=....
```

DELETE all user files

```
DELETE /api/files/delete-all
```

User: manage user settings

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 service configuration

```
GET /api/service/config
```

GET job service state

```
GET /api/service/state
```

GET disk usage state

```
GET /api/service/disk-use
```

POST refresh disk space usage info

```
POST /api/service/disk-use/refersh
```

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
DELETE /api/service/job/delete/history-all/:success
```

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 close model database

```
POST /api/admin/model/:model/close
```

POST a request to delete the model

```
POST /api/admin/model/:model/delete
```

POST a request to open database file

```
POST /api/admin/db-file-open/:path
```

POST a request to cleanup database file

```
POST /api/admin/db-cleanup/:path
POST /api/admin/db-cleanup/:path/name/:name
POST /api/admin/db-cleanup/:path/name/:name/digest/:digest
```

GET the list of database cleanup logs

```
GET /api/admin/db-cleanup/log-all
```

GET database cleanup log file

```
GET /api/admin/db-cleanup/log/:name
```

POST a request to pause model run queue

```
POST /api/admin/jobs-pause/:pause
```

POST a request to pause all queues of model runs

```
POST /api/admin-all/jobs-pause/:pause
```

PUT a request to shutdown web-service

```
PUT /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...
            disk.ini  -> (optional) disk usage control settings to set storage quotas
   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
        doc/  -> models documentation directory
        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
        past/     -> (optional) shadow copy of history folder, invisible to the end user
        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
            jobs.queue.all.paused  -> if such file exists then all jobs in all queues are 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
MpiCpu        = 40  ; max MPI cpu cores available for each oms instance, zero means oms instances can use all cpu's available
MpiMemory     = 0   ; gigabytes, max MPI memory available for each oms instance, zero means oms instances can use all memory available
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...
            disk.ini  -> (optional) disk usage control settings to set storage quotas for Bob and Alice
    etc/     -> config files directory, contain template(s) to run models
    alice/   -> user Alice "root" directory
        log/     -> recommended Alice's log files directory
        models/
            bin/  -> Alice's model.exe and model.sqlite directory
            log/  -> Alice's directory for models run log files
            doc/  -> models documentation directory
            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
        log/     -> recommended Bob's log files directory
        models/
            bin/  -> Bob's model.exe and model.sqlite directory
            log/  -> Bob's directory for models run log files
            doc/  -> models documentation directory
            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
        past/     -> (optional) shadow copy of history folder, invisible to the end user
        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
            jobs.queue.all.paused  -> if such file exists then all jobs in all queues are 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 &'
```

Cloud disks usage: limit storage space usage

You may want to set the limits on disk space usage and enforce storage cleanup by users. It can be done through `etc/disk.ini` file. If `etc/disk.ini` exists then `oms` web-service will monitor and report disk usage by user(s) and may set a limit on storage space. You can set a limit for individual user, group of users and grand total space limit on storage space used by all users. If user exceeding disk space quotas then she/he cannot run the model or upload files to cloud, only download is available. User can Cleanup Disk Space through UI.

Example of `disk.ini`:

```
; Example of storage usage control settings
;   "user" term below means oms instance
;   "user name" is oms instance name, for example: "localhost_4040"
;
; if etc/disk.ini file exists then storage usage control is active

[Common]

; seconds, storage scan interval, if too small then default value used
;
ScanInterval  =   0

; GBytes, user storage quota, default: 0 (unlimited)
;
UserLimit     =   0

; GBytes, total storage quota for all users, default: 0 (unlimited)
;   if non-zero then it restricts the total storage size of all users
;
AllUsersLimit = 128

; Database cleanup script:
;   creates new model.sqlite database and copy model data
;
DbCleanup = etc/db-cleanup_linux.sh

; user groups can be created to simplify settings
;
Groups = Low, High, Others

[Low]
Users      = localhost_4040, bob, alice
UserLimit  = 2

[High]
Users      = king, boss, cheif
UserLimit  = 20

[king]
UserLimit  = 100 ; override storage settings for oms instance "king"

; "me" is not a member of any group
;
[me]
UserLimit  = 0 ; unlimited
```

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, if empty then server is always ready, no startup
StopExe   = /bin/bash             ; default executable to stop server, if empty then server is always ready, no shutdown
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 stop 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, if empty then server is always ready, no startup
StopExe   = cmd                       ; default executable to stop server, if empty then server is always ready, no shutdown
StartArgs = /C-@-etc\compute-start.bat  ; default command line arguments to start server, server name will be appended
StopArgs  = /C-@-etc\compute-stop.bat   ; default command line arguments to stop 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, if empty then server is always ready, no startup
StopExe   = /bin/bash                    ; default executable to stop server, if empty then server is always ready, no shutdown
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 stop 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
            disk.ini  -> (optional) disk usage control settings to set storage quotas Bob and Alice
    log/     -> recommended log files directory
    alice/   -> user Alice "root" directory
        log/     -> recommended Alice's log files directory
        models/
            bin/  -> Alice's model.exe and model.sqlite directory
            log/  -> Alice's directory for models run log files
            doc/  -> models documentation directory
            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
        log/     -> recommended Bob's log files directory
        models/
            bin/  -> Bob's model.exe and model.sqlite directory
            log/  -> Bob's directory for models run log files
            doc/  -> models documentation directory
            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
        past/     -> (optional) shadow copy of history folder, invisible to the end user
        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.

There is also an excelent R package created by Matthew T. Warkentin available at: mattwarkentin.github.io/openmpp.

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
#
# If any of library below is not installed then do:
#   install.packages("jsonlite")
#   install.packages("httr")
#
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.

There is also an excelent R package created by Matthew T. Warkentin available at: mattwarkentin.github.io/openmpp.

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

Input Excel workbook:

Output Excel workbook:

ModelName	ModelVersion	RunName	SubCount	RunStarted	RunCompleted	RunDescription	RunNotes
RiskPaths	3.0.0.0	New 123,000 cases	8	2023-02-07 14:28:42.474	2023-02-07 14:28:43.904	New scenario with 123,000 simulation cases	
RiskPaths	3.0.0.0	New 456,000 cases	8	2023-02-07 14:31:07.492	2023-02-07 14:31:11.482	New scenario with 456,000 simulation cases	
RiskPaths	3.0.0.0	New 789,000 cases	8	2023-02-07 14:32:47.607	2023-02-07 14:32:55.174	New scenario with 789,000 simulation cases	

RunInfo | TableInfo | T04_FertilityRatesByAgeGroup | T03_FertilityByAge

output-tables-data.xlsx - Excel

A1: TableName

TableName	TableDescription	TableNotes
T03_FertilityByAge	Age-specific fertility	
T04_FertilityRatesByAgeGroup	Fertility rates by age group	

RunInfo | TableInfo | T04_FertilityRatesByAgeGroup | T03_FertilityByAge

output-tables-data.xlsx - Excel

A1: expr_name

expr_name	Dim0	Dim1	New 123,000 cases	New 456,000 cases	New 789,000 cases
Expr0	(-∞,15)	US_NEVER_IN_UNION	0	0	0
Expr0	(-∞,15)	US_FIRST_UNION_PERIOD1	null	null	null
Expr0	(-∞,15)	US_FIRST_UNION_PERIOD2	null	null	null
Expr0	(-∞,15)	US_AFTER_FIRST_UNION	null	null	null
Expr0	(-∞,15)	US_SECOND_UNION	null	null	null
Expr0	(-∞,15)	US_AFTER_SECOND_UNION	null	null	null
Expr0	[15,17.5)	US_NEVER_IN_UNION	0.0186844356489197	0.0186801447393292	0.0187801691158234
Expr0	[15,17.5)	US_FIRST_UNION_PERIOD1	0.28236266934192	0.283948359054652	0.286120080376247

RunInfo | TableInfo | T04_FertilityRatesByAgeGroup | T03_FertilityByAge

output-tables-data.xlsx - Excel

A1: expr_name

expr_name	Dim0	New 123,000 cases	New 456,000 cases	New 789,000 cases
Expr0	(-∞,15)	0	0	0
Expr0	[15,16)	0.021439024	0.021760965	0.021996198
Expr0	[16,17)	0.027691057	0.027758772	0.027922687
Expr0	[17,18)	0.061837398	0.061559211	0.061588086
Expr0	[18,19)	0.109138211	0.108199561	0.107726236
Expr0	[19,20)	0.108260163	0.109344298	0.109187579
Expr0	[20,21)	0.114894309	0.114269737	0.113614702
Expr0	[21,22)	0.098121951	0.098432018	0.09861597

TableInfo | T04_FertilityRatesByAgeGroup | T03_FertilityByAge

R script

```r
#
# Read multiple tables from multiple model runs and save it as XLSX file
# Also save model runs metadata and tables metadata (name, description, notes) into .csv files
# Model run names and table names are coming from another input XLSX file
#
# If any of library below is not installed then do:
#   install.packages("jsonlite")
#   install.packages("httr")
#   install.packages("readxl")
#   install.packages("writexl")
#
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
#

shts <- list(
    RunInfo = runInfo,
    TableInfo = tableInfo
  )

for (tbl in tn$TableNames)
{

  allCct <- NULL
  isFirst <- TRUE

  for (run in rn$RunNames)
  {
    cct <- read.csv(paste0(
      apiUrl, "model/", md, "/run/", URLencode(run, reserved = TRUE), "/table/", tbl, "/expr/csv"
      ))

    # build a pivot table data frame:
    # use first run results to assign all dimensions and measure(s)
    # from all subsequent model run bind only expr_value column
    if (isFirst) {
      allCct <- rbind(allCct, cct)
      isFirst <- FALSE
    } else {
      cval <- data.frame(expr_value = cct$expr_value)
      allCct <- cbind(allCct, cval)
    }

    # use run name for expression values column name
    names(allCct)[names(allCct) == 'expr_value'] <- run
  }
  shts[[ tbl ]] <- allCct
}

write_xlsx(shts, paste0("output-tables-data.xlsx"))
```

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.

There is also an excelent R package created by Matthew T. Warkentin available at: mattwarkentin.github.io/openmpp.

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 3.6.1.5
#   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")
```

```r
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")

#
# Model digest of OncoSimX-lung version 3.6.1.5: "eeb246bd7d3bdb64d3e7aaefeaa828ea"
#
md <- "eeb246bd7d3bdb64d3e7aaefeaa828ea"

# 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 ....
#
# It is a sequential run, not parallel.
#
# Running 4 OncoSimX-lung_mpi instances: "root" leader process and 3 computational processes
# each computational process using modelling 4 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
  # use explicit model run stamp to avoid compatibility issues between cloud model run queue and desktop MPI
  stamp <- sub('.' , '_', fixed = TRUE, format(Sys.time(),"%Y_%m_%d_%H_%M_%OS3"))

  # prepare model run options
  pd <- list(
      ModelDigest = md,
      Mpi = list(
        Np = 4,              # MPI cluster: run 4 processes: 3 for model and rott process
        IsNotOnRoot = TRUE    # MPI cluster: do not use root process for modelling
      ),
      Template = "mpi.OncoSimX.template.txt",  # MPI cluster: model run template
      Opts = list(
        Parameter.LcScreenSmokingDurationCriteria = toString(smokingDuration[k]),
        Parameter.SimulationCases = "6000",    # use only 6000 simulation cases for quick test
        OpenM.BaseRunDigest = firstRunDigest,  # base run to get the rest of input parameters
        OpenM.SubValues = "12",                # use 12 sub-values (sub-samples)
        OpenM.Threads = "4",                   # use 4 modeling threads
        OpenM.RunStamp = stamp,                # use explicit run stamp
         # 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)
  submitStamp <- jr$SubmitStamp # model run submission stamp: not empty if model run submitted to run on cluster
  runStamp <- jr$RunStamp       # model run stamp: not empty if model run started

  # wait until model run completed
  runDigests[k] <- waitForRunCompleted(stamp, 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 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.

There is also an excelent R package created by Matthew T. Warkentin available at: mattwarkentin.github.io/openmpp.

Below is an example of `oms` JSON web-service usage to run RiskPaths model on cloud grid from RStudio in cloud. There is an identical example to:

- run RiskPaths model in cloud from local PC Rstudio
- 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
```

```
#   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)
```

```r
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 a sequential run, not parallel.
#
# Running 4 RiskPaths_mpi instances: "root" leader process and 3 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))

# use explicit model run stamp to avoid compatibility issues between cloud model run queue and desktop MPI
stamp <- sub('.' , '_', fixed = TRUE, format(Sys.time(),"%Y_%m_%d_%H_%M_%OS3"))

# prepare model run options
pd <- list(
    ModelDigest = md,
    Mpi = list(
      Np = 5,              # MPI cluster: run 5 processes: 4 for model and rott process
      IsNotOnRoot = TRUE    # MPI cluster: do not use root process for modelling
    ),
    Template = "mpi.RiskPaths.template.txt",  # MPI cluster: model run tempate
    Opts = list(
      OpenM.TaskName = taskName,
      OpenM.RunStamp = stamp,               # use explicit run stamp
      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)

submitStamp <- jr$SubmitStamp # model run submission stamp: not empty if model run submitted to run queue
runStamp <- jr$RunStamp       # model run stamp: by default empty until model run not started

# wait until task completed
runDigests <- waitForTaskCompleted(taskName, stamp, apiUrl, md)

#
# get results of task run, cohort fertility: T05_CohortFertility.Expr1
#
```

```r
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 (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 RiskPaths model in cloud from local PC

OpenM++ integration with R: use local PC RStudio to 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.

There is also an excelent R package created by Matthew T. Warkentin available at: mattwarkentin.github.io/openmpp.

Below is an example of `oms` JSON web-service usage to run RiskPaths model on cloud grid from your local PC RStudio. There is an identical example to:

- run RiskPaths model in cloud
- 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.

Prerequisite Following environmemnt variables are required:

```
OM_CLOUD_URL=https://your-user.cloud.org
OM_CLOUD_USER=your-login-name
OM_CLOUD_PWD=your-secret-password
```

If there are no any other options avaliable then you can store above values in `.Renviron` file in your `HOME` directory. On Windows HOME directory is: "C:\Users\User Name Here\Documents". **Important Security Warning:** `.Renviron` file is NOT a safe place to strore login information. Contact your IT security team for better solution.

Important: Clear console and clear history after checking your login name and password.

R script

```
#
# R integration example using RiskPaths model
#   to analyze contribution of delayed union formations
#   versus decreased fertility on childlessness
#
# Cloud model run from local user PC
#
# Prerequisite:
#
# 1.
# Cloud user account:
#   Following environmemnt variables are required:
#     OM_CLOUD_URL  - cloud URL, e.g.:       https://model.openmpp.org
#     OM_CLOUD_USER - user login name, e.g.: demo
#     OM_CLOUD_PWD  - login password, e.g.:  my-secret-password
#
# You can use .Renviron file to define it if there are no any other options avaliable.
# !!! Security warning:
#     .Renviron file is not the safe place to store passwords
#
# Script below is using openM++ web-service "oms" in cloud
# to run the model, modify parameters and read output values.
#
# 2.
# omsCommon.R file which contains helper functions.
#
# Place it in 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

# 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
#
source("~/omsCommon.R")

# login to cloud workspace
#
lg <- loginToOpenmCloud()
apiUrl <- lg$apiUrl
loginToken <- lg$loginToken

#
# 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"
```

```
#  ModelDigest = d90e1e9a49a00d972ec1d30e084002b
#
md <- "RiskPaths"


# 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"
    ),
    set_cookies(jwt_token = loginToken)
  )
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"
    ),
    set_cookies(jwt_token = loginToken)
  )
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"
    ),
    set_cookies(jwt_token = loginToken)
  )
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(),
        set_cookies(jwt_token = loginToken)
      )
    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(),
    set_cookies(jwt_token = loginToken)
  )
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 a sequential run, not parallel.
#
# Running 4 RiskPaths_mpi instances: "root" leader process and 3 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))

# use explicit model run stamp to avoid compatibility issues between cloud model run queue and desktop MPI
stamp <- sub('.' , '_', fixed = TRUE, format(Sys.time(),"%Y_%m_%d_%H_%M_%OS3"))

# prepare model run options
pd <- list(
    ModelDigest = md,
    Mpi = list(
      Np = 5,              # MPI cluster: run 5 processes: 4 for model and rott process
      IsNotOnRoot = TRUE    # MPI cluster: do not use root process for modelling
    ),
    Template = "mpi.RiskPaths.template.txt",  # MPI cluster: model run tempate
    Opts = list(
      OpenM.TaskName = taskName,
      OpenM.RunStamp = stamp,               # use explicit run stamp
      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(),
    set_cookies(jwt_token = loginToken)
  )
if (http_type(rsp) != 'application/json') {
  stop("Failed to run the model")
}
jr <- content(rsp)

submitStamp <- jr$SubmitStamp # model run submission stamp: not empty if model run submitted to run queue
runStamp <- jr$RunStamp       # model run stamp: by default empty until model run not started

# wait until task completed
runDigests <- waitForTaskCompleted(taskName, stamp, apiUrl, md, loginToken)

#
# 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 (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(),
        set_cookies(jwt_token = loginToken)
      )
    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
  }
}
```

```r
#
# 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
)

# Cleanup:
# delete modelling task
# delete all input scenarios

print(paste("Delete", taskName))

rsp <- DELETE(
    paste0(
      apiUrl, "model/", md, "/task/", taskName
    ),
    set_cookies(jwt_token = loginToken)
  )
stop_for_status(rsp, "delete modelling task")

for (sn in nameLst)
{
  print(paste("Delete", sn))

  rsp <- POST(
      paste0(
        apiUrl, "model/", md, "/workset/", sn, "/readonly/false"
      ),
      set_cookies(jwt_token = loginToken)
    )
  stop_for_status(rsp, paste("update read-only status of input set", sn))

  rsp <- DELETE(
      paste0(
        apiUrl, "model/", md, "/workset/", sn
      ),
      set_cookies(jwt_token = loginToken)
    )
  stop_for_status(rsp, paste("delete input set", sn))
}
```

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.

There is also an excelent R package created by Matthew T. Warkentin available at: mattwarkentin.github.io/openmpp.

Below is an example of `oms` JSON web-service usage to run RiskPaths model on cloud grid from RStudio in cloud. There is an identical example to:

- run RiskPaths model in cloud from local PC Rstudio
- 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
```

```
#   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)
```

```r
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 a sequential run, not parallel.
#
# Running 4 RiskPaths_mpi instances: "root" leader process and 3 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))

# use explicit model run stamp to avoid compatibility issues between cloud model run queue and desktop MPI
stamp <- sub('.' , '_', fixed = TRUE, format(Sys.time(),"%Y_%m_%d_%H_%M_%OS3"))

# prepare model run options
pd <- list(
    ModelDigest = md,
    Mpi = list(
      Np = 5,              # MPI cluster: run 5 processes: 4 for model and rott process
      IsNotOnRoot = TRUE    # MPI cluster: do not use root process for modelling
    ),
    Template = "mpi.RiskPaths.template.txt",  # MPI cluster: model run tempate
    Opts = list(
      OpenM.TaskName = taskName,
      OpenM.RunStamp = stamp,               # use explicit run stamp
      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)

submitStamp <- jr$SubmitStamp # model run submission stamp: not empty if model run submitted to run queue
runStamp <- jr$RunStamp       # model run stamp: by default empty until model run not started

# wait until task completed
runDigests <- waitForTaskCompleted(taskName, stamp, apiUrl, md)

#
# get results of task run, cohort fertility: T05_CohortFertility.Expr1
#
```

```r
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 (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: 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

There is also an excelent R package created by Matthew T. Warkentin available at: mattwarkentin.github.io/openmpp.

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 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 to:

- run RiskPaths model in cloud
- run RiskPaths model in cloud from local PC Rstudio

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

There is also an excelent R package created by Matthew T. Warkentin available at: mattwarkentin.github.io/openmpp.

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/omnp/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 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
# 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.
# It is the same as command line:
#   NewCaseBased.exe -OpenM.ProgressPercent 100 -Parameter.MortalityHazard 0.014
#   NewCaseBased.exe -OpenM.ProgressPercent 100 -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',
            'OpenM.SubValues': '16',
            'OpenM.Threads': '4'
        }
```

Ln: 1 Col: 0

Python script

```python
#
# 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
```

```python
#
# 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[n], "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:
```

```
#     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()
```

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":

This menu can be accessed by right-clicking the Docker icon in the system tray, which is normally in the lower right corner of the display.

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:

- 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 stable (12) 11 and 10, MX Linux 23, 21 and 19, Ubuntu 22.04, RedHat 9+
- 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)

Note: It does work on most of latest Linux distributions, we just not testing regularly on every Linux version.

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 20.04, Ubuntu 18.04 and RedHat 8.

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
```

Note: Above output does not include all possible Linux versions and may be outdated, openM++ supports almost any of modern Linux distribution.

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 11, 10, it may also work on 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 2022 or 2019 (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 latest, may work starting from Catalina 10.15 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/go-ompp
  cd $HOME/go-ompp
  export GOPATH=$HOME/go-ompp
  git clone https://github.com/openmpp/go ompp-go
  ```

- build Go utilities:

  ```
  cd $HOME/go-ompp/ompp-go
  go install -tags sqlite_math_functions,sqlite_omit_load_extension ./dbcopy
  go install -tags sqlite_math_functions,sqlite_omit_load_extension ./oms
  ```

After initial checkout first `go install` command can take ~30 seconds because go needs to get all dependencies.

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,sqlite_math_functions,sqlite_omit_load_extension ./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 stable (12) 11 and 10, MX Linux 23, 21 and 19, Ubuntu 22.04, RedHat 9+, (Ubuntu 20.04 and RedHat 8 may also work but not tested regualry)
  - Windows (tested): 11, 10, may work on 7
  - tested on MacOS latest, may work starting from Catalina 10.15 and Big Sur 11.1+, 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 2022, including Community Edition, Visual Studio 2019 also works, but not tested regularly
  - 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+
- Go 1.19+, 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
```

Note: Above output does not include all possible Linux versions and may be outdated, openM++ supports latest Linux distributions.

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 2022 or 2019 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:

```
#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:

```
#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:

```
#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:

```
# .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
```

Command Prompt

Windows 32bit

WizardCaseBased model, 8 sub-samples

```
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:         nednodes=64:ppn=8,nodes=64:ppn=8,walltime=00:15:00
Resources:      cput=24:53:50,mem=24381388kb,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

```
C:\om\prototype_lib\openm\Release\x64>
C:\om\prototype_lib\openm\Release\x64>mpiexec -np 8 Alpha1 -General.Cases 10000000
2013-09-26 11:08:31.0779 Model: Alpha1
2013-09-26 11:08:31.0779 Model: Alpha1
2013-09-26 11:08:31.0779 Model: Alpha1
2013-09-26 11:08:31.0779 Model: Alpha1
2013-09-26 11:08:31.0779 Model: Alpha1
2013-09-26 11:08:31.0779 Model: Alpha1
2013-09-26 11:08:31.0779 Model: Alpha1
2013-09-26 11:08:31.0779 Model: Alpha1
2013-09-26 11:08:31.0857 Reading Parameters
2013-09-26 11:08:31.0857 Running Simulation
2013-09-26 11:08:31.0857 Reading Parameters
2013-09-26 11:08:31.0857 Reading Parameters
2013-09-26 11:08:31.0857 Running Simulation
2013-09-26 11:08:31.0857 Running Simulation
2013-09-26 11:08:31.0857 Reading Parameters
2013-09-26 11:08:31.0857 Reading Parameters
2013-09-26 11:08:31.0857 Running Simulation
2013-09-26 11:08:31.0857 Running Simulation
2013-09-26 11:08:31.0904 Reading Parameters
2013-09-26 11:08:31.0904 Running Simulation
2013-09-26 11:08:31.0997 Reading Parameters
2013-09-26 11:08:31.0997 Running Simulation
2013-09-26 11:08:31.0169 Reading Parameters
2013-09-26 11:08:31.0169 Running Simulation
2013-09-26 11:11:15.0158 Writing Output Tables
2013-09-26 11:11:16.0188 Writing Output Tables
2013-09-26 11:11:17.0874 Writing Output Tables
2013-09-26 11:11:38.0779 Writing Output Tables
2013-09-26 11:11:55.0485 Writing Output Tables
2013-09-26 11:12:06.0999 Writing Output Tables
2013-09-26 11:12:10.0274 Writing Output Tables
2013-09-26 11:12:12.0005 Writing Output Tables
2013-09-26 11:12:12.0130 Done.
2013-09-26 11:12:12.0130 Done.
2013-09-26 11:12:12.0130 Done.
2013-09-26 11:12:12.0130 Done.
2013-09-26 11:12:12.0130 Done.
2013-09-26 11:12:12.0130 Done.
2013-09-26 11:12:12.0130 Done.
2013-09-26 11:12:12.0130 Done.

C:\om\prototype_lib\openm\Release\x64>\sqlite\sqlite3 -header -column Alpha1.sqlite "SELEC
T S.unit_id AS "Dim0", S.value AS "Value" FROM alpha1_201309261008330703_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
```

Command Prompt

```
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_20130926100833070 3_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
```

Windows 32bit

Alpha1 model
8 subsamples
10,000,000 cases

mpi@omm:~

```
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
```

Beyond2020 Linux HPC cluster

64bit

Alpha1 model
8 subsamples
10,000,000 cases

...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
_20130926132811085 5_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.u
nit_id AS "Dim0", S.value AS "Value" FROM alpha1_201309271035060599_v0_DurationOfLife S WHERE S.run_id = ( SE
LECT MAX(RL.run_id) FROM run_lst RL INNER JOIN model_dic MD ON (MD.model_id = RL.model_id) WHERE MD.model_nam
e = '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=85000808kb,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_d
ic 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.

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
OM_CFG_TYPE_MAX_LEN=366
```

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
- `OM_CFG_TYPE_MAX_LEN` : type size limit, if dimension type exceed this size then UI would not show parameter or output table with that dimension

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 oms root directory, default: models/bin
-oms.ModelLogDir:    models log directory, if relative then must be relative to oms root directory: default: models/log
-oms.ModelDocDir:    models documentation directory, if relative then must be relative to oms root directory: default: models/doc
-oms.HomeDir:        user personal home directory, if relative then must be relative to oms 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.FilesDir:       user files directory, if relative then must be relative to oms root directory, if home directory specified then it is: home/io
-oms.AllowMicrodata: if true then allow model run microdata
-oms.HtmlDi:         front-end UI directory, if relative then must be relative to oms 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 oms 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 oms 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.AdminAll        if true then allow global administrative routes: /admin-all/
-oms.NoAdmin         if true then disable local administrative routes: /admin/
-oms.NoShutdown      if true then disable shutdown route: /shutdown/

-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
# This is also a comment

; Ini file can be supplied to oms.exe as command line option "-ini" or "-OpenM.IniFile"
; "-ini" is a short form of "-OpenM.IniFile", command lines below are equal:
;
# oms.exe -ini           path/to/oms.ini
# oms.exe -OpenM.IniFile path/to/oms.ini

; "-l"   is a short form of "-oms.Listen", command lines below are equal:
;
# oms.exe -l          localhost:4040
# oms.exe -oms.Listen localhost:4040

; boolean options can be "true" or "false" or empty value
; boolean empty value is the same as "true"
; for example both command lines below are equal:
;
# oms -oms.ApiOnly
# oms -oms.ApiOnly true

[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
; ModelDocDir    = models/doc     # models documentation directory, default: models/doc, 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
; FilesDir       =                # user files directory, if relative then must be relative to oms root directory, if home directory specified then it is: home/io
; AllowMicrodata = false          # if true then allow model run microdata
; 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
; 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
; Languages      = en             # comma-separated list of supported languages
; CodePage       =                # code page to convert source file into utf-8, e.g.: windows-1252
; DoubleFormat   = %.15g          # format to convert float or double value to string, e.g. %.15g
; AdminAll       = false          # if true then allow global administrative routes: /admin-all/
; NoAdmin        = false          # if true then disable local administrative routes: /admin/
; NoShutdown     = false          # if true then disable shutdown route: /shutdown/

[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

; "-v" is a short form of "-OpenM.LogToConsole"

; 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 pid-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
```

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
        doc/  -> models documentation directory
```

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 -oms.ModelDocDir /my-models/doc2
```

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, model ini-files, downloads of model model run results or upload input scenarios:

```
oms -oms.HomeDir models/home -oms.AllowDownload
oms -oms.HomeDir models/home -oms.AllowDownload -oms.AllowUpload
```

Important: Please keep in mind, if you allow users to download or upload files on your server then it may create a security risk.

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
        doc/  -> models documentation directory
        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
             disk.ini  -> (optional) disk usage control settings to set storage quotas
    log/     -> recommended log files directory
    models/
        bin/  -> default model.exe and model.sqlite directory
        log/  -> default directory for models run log files
        doc/  -> models documentation directory
        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 to restrict storage size
        active/   -> active model run state files
        history/  -> model run history files
        past/     -> (optional) shadow copy of history folder, invisible to the end user
        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
            jobs.queue.all.paused  -> if such file exists then all jobs in all queues are 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 /api/model/:model/pack
```

GET model metadata including text (description and notes)

```
GET /api/model/:model/text
GET /api/model/:model/text/lang/:lang
GET /api/model/:model/pack/text
GET /api/model/:model/pack/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 output table calculated values from model run

```
POST /api/model/:model/run/:run/table/calc
```

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

```
POST /api/model/:model/run/:run/table/calc-id
```

Read output table values and compare model runs

```
POST /api/model/:model/run/:run/table/compare
```

Read output table values and compare model runs (enun id's)

```
POST /api/model/:model/run/:run/table/compare-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
```

Read aggregated microdata from model run

```
POST /api/model/:model/run/:run/microdata/calc
```

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

```
POST /api/model/:model/run/:run/microdata/calc-id
```

Read microdata run comparison

```
POST /api/model/:model/run/:run/microdata/compare
```

Read microdata run comparison (enum id's)

```
POST /api/model/:model/run/:run/microdata/compare-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 calculated expression(s) from model run

```
GET /api/model/:model/run/:run/table/:name/calc/:calc
GET /api/model/:model/run/:run/table/:name/calc/:calc/start/:start
GET /api/model/:model/run/:run/table/:name/calc/:calc/start/:start/count/:count
```

GET output table values and compare model runs

```
GET /api/model/:model/run/:run/table/:name/compare/:compare/variant/:variant
GET /api/model/:model/run/:run/table/:name/compare/:compare/variant/:variant/start/:start
GET /api/model/:model/run/:run/table/:name/compare/:compare/variant/:variant/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 aggregated microdata from model run

```
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/calc/:calc
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/calc/:calc/start/:start
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/calc/:calc/start/:start/count/:count
```

GET microdata run comparison

```
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/compare/:compare/variant/:variant
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/compare/:compare/variant/:variant/start/:start
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/compare/:compare/variant/:variant/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 calculated table expressions from model run

```
GET /api/model/:model/run/:run/table/:name/calc/:calc/csv
GET /api/model/:model/run/:run/table/:name/calc/:calc/csv-bom
```

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

```
GET /api/model/:model/run/:run/table/:name/calc/:calc/csv-id
GET /api/model/:model/run/:run/table/:name/calc/:calc/csv-id-bom
```

GET csv model runs comparison table expressions

```
GET /api/model/:model/run/:run/table/:name/compare/:compare/variant/:variant/csv
GET /api/model/:model/run/:run/table/:name/compare/:compare/variant/:variant/csv-bom
```

GET csv model runs comparison table expressions (enum id's)

```
GET /api/model/:model/run/:run/table/:name/compare/:compare/variant/:variant/csv-id
GET /api/model/:model/run/:run/table/:name/compare/:compare/variant/:variant/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 csv aggregated microdata from model run

```
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/calc/:calc/csv
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/calc/:calc/csv-bom
```

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

```
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/calc/:calc/csv-id
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/calc/:calc/csv-id-bom
```

GET csv microdata run comparison

```
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/compare/:compare/variant/:variant/csv
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/compare/:compare/variant/:variant/csv-bom
```

GET csv microdata run comparison (enum id's)

```
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/compare/:compare/variant/:variant/csv-id
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/compare/:compare/variant/:variant/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
```

POST delete multiple worksets

```
POST /api/model/:model/delete-worksets
```

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
```

POST delete model runs

```
POST /api/model/:model/delete-runs
```

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 /api/download/start/delete/:folder
```

DELETE all download files

```
DELETE /api/download/delete-all
DELETE /api/download/start/delete-all
```

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 /api/upload/start/delete/:folder
```

DELETE all upload files

```
DELETE /api/upload/delete-all
DELETE /api/upload/start/delete-all
```

Download and upload user files

GET user files tree

```
GET /api/files/file-tree/:ext/path/:path
GET /api/files/file-tree/:ext/path/
GET /api/files/file-tree/:ext/path
GET /api/files/file-tree/:ext/path?path=....
```

POST upload to user files

```
POST /api/files/file/:path
POST /api/files/file?path=....
```

PUT create user files folder

```
PUT /api/files/folder/:path
PUT /api/files/folder?path=...
```

DELETE file or folder from user files

```
DELETE /api/files/delete/:path
DELETE /api/files/delete?path=....
```

DELETE all user files

```
DELETE /api/files/delete-all
```

User: manage user settings

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 service configuration

```
GET /api/service/config
```

GET job service state

```
GET /api/service/state
```

GET disk usage state

```
GET /api/service/disk-use
```

POST refresh disk space usage info

```
POST /api/service/disk-use/refersh
```

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
DELETE /api/service/job/delete/history-all/:success
```

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 close model database

```
POST /api/admin/model/:model/close
```

POST a request to delete the model

```
POST /api/admin/model/:model/delete
```

POST a request to open database file

```
POST /api/admin/db-file-open/:path
```

POST a request to cleanup database file

```
POST /api/admin/db-cleanup/:path
POST /api/admin/db-cleanup/:path/name/:name
POST /api/admin/db-cleanup/:path/name/:name/digest/:digest
```

GET the list of database cleanup logs

```
GET /api/admin/db-cleanup/log-all
```

GET database cleanup log file

```
GET /api/admin/db-cleanup/log/:name
```

POST a request to pause model run queue

```
POST /api/admin/jobs-pause/:pause
```

POST a request to pause all queues of model runs

```
POST /api/admin-all/jobs-pause/:pause
```

PUT a request to shutdown web-service

```
PUT /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.*


```
[
  {
    "Model": {
      "ModelId": 101,
      "Name": "HPVMM",
      "Digest": "0636a04c5462b376b652132196095c22",
      "Type": 1,
      "Version": "1.9.2.0",
      "CreateDateTime": "2023-03-20 11:02:11.404",
      "DefaultLangCode": "EN"
    },
    "Dir": "HPVMM-1.9.2.0",
    "DbPath": "HPVMM-1.9.2.0/HPVMM.sqlite"
  },
  {
    "Model": {
      "ModelId": 101,
      "Name": "RiskPaths",
      "Digest": "d90e1e9a49a06d972ecf1d50e684c62b",
      "Type": 0,
      "Version": "3.0.0.0",
      "CreateDateTime": "2024-02-16 20:32:48.521",
      "DefaultLangCode": "EN"
    },
    "Dir": ".",
    "DbPath": "RiskPaths.sqlite"
  }
]
```

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
GET /api/model/:model/pack
```

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
http://localhost:4040/api/model/RiskPaths/pack
```

The `/pack/` version of the method produce more compact output results. For example range types are retrurned as `[min, max]` enum id's only:


```
{
    "ModelId": 101,
    "TypeId": 103,
    "TypeHid": 101,
    "Name": "LIFE",
    "Digest": "90bc3191bed66b07d1c7b9f88c42584c",
    "DicId": 3,
    "TotalEnumId": 101,
    "IsRange": true,
    "MinEnumId": 0,
    "MaxEnumId": 100,
    "Enum": null
},
```


And a regular method (without `/pack/`) return a full list of enum type values:


```
{
    "ModelId": 101,
    "TypeId": 103,
    "TypeHid": 101,
    "Name": "LIFE",
    "Digest": "90bc3191bed66b07d1c7b9f88c42584c",
    "DicId": 3,
    "TotalEnumId": 101,
    "IsRange": true,
    "MinEnumId": 0,
    "MaxEnumId": 100,
    "Enum": [{
        "ModelId": 101,
        "TypeId": 103,
        "EnumId": 0,
        "Name": "0"
      },
  ................
  ................
      {
        "ModelId": 101,
        "TypeId": 103,
        "EnumId": 100,
        "Name": "100"
      }
    ]
}
```


Return example:


```
{
  "Model": {
    "ModelId": 1,
    "Name": "modelOne"
```

```
    "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,
      "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",
"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 A
ND 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,
```

"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.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,
        "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
GET /api/model/:model/pack/text
GET /api/model/:model/pack/text/lang/:lang
```

The `/pack/` version of the method produce more compact output results. For example range types are retrurned as `[min, max]` enum id's only:


```
{
    "Type": {
      "ModelId": 101,
      "TypeId": 103,
      "TypeHid": 101,
      "Name": "LIFE",
      "Digest": "90bc3191bed66b07d1c7b9f88c42584c",
      "DicId": 3,
      "TotalEnumId": 101,
      "IsRange": true,
      "MinEnumId": 0,
      "MaxEnumId": 100
    },
    "DescrNote": {
      "LangCode": "EN",
      "Descr": "Simulated age range",
      "Note": ""
    },
    "TypeEnumTxt": []
},
```


And a regular method (without `/pack/`) return a full list of enum type values:

```
{
    "Type": {
      "ModelId": 101,
      "TypeId": 103,
      "TypeHid": 101,
      "Name": "LIFE",
      "Digest": "90bc3191bed66b07d1c7b9f88c42584c",
      "DicId": 3,
      "TotalEnumId": 101,
      "IsRange": true,
      "MinEnumId": 0,
      "MaxEnumId": 100
    },
    "DescrNote": {
      "LangCode": "EN",
      "Descr": "Simulated age range",
      "Note": ""
    },
    "TypeEnumTxt": [{
        "Enum": {
          "ModelId": 101,
          "TypeId": 103,
          "EnumId": 0,
          "Name": "0"
        },
        "DescrNote": {
          "LangCode": "EN",
          "Descr": "0",
          "Note": ""
        }
      }, {
  ................
  ................
      }, {
        "Enum": {
          "ModelId": 101,
          "TypeId": 103,
          "EnumId": 100,
          "Name": "100"
        },
        "DescrNote": {
          "LangCode": "EN",
          "Descr": "100",
          "Note": ""
        }
      }
    ]
}
```


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
http://localhost:4040/api/model/RiskPaths/pack/text
```

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": {
```

```
      "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": {
    "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,
```

```
        "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",
```

```
    "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,
```

```
      "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": [],
```

```
    "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 00:28:01.058",
```

"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,
  "IsFullPage": true,
  "IsSubId": true,
  "SubId": 2,
  "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, if size <= 0 then all rows selected
IsFullPage - (optional) if true then always return non-empty last page of data
IsSubId    - (optional) if true then select only single sub-value, default: all sub-values
SubId      - (optional) sub-value id to select if IsSubId is true
Filter     - (optional) conditions to filter dimension enum code(s)
OrderBy    - (optional) list of columns indexes (one based) to order by
```

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,"IsFullPage":false,"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,
  "IsFullPage": true,
  "Filter":     [],
  "OrderBy":    []
}
```

Result:


```
{"Layout":{"Offset":6,"Size":2,"IsFullPage":true,"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,"IsFullPage":false,"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": "isOldAge",
  "Offset": 0,
  "Size": 0,
  "Filter": [{
      "Name": "dim0",
      "Op": "IN",
      "Values": ["20-30", "40+"]
    }
  ],
  "OrderBy": [{
      "IndexOne": 2,
      "IsDesc": true
    }
  ],
  "IsSubId": true,
  "SubId": 2
}
```

Result:

{
"Page":[
{"Dims":["40+"],"IsNull":false,"Value":true,"SubId":2}
,{"Dims":["20-30"],"IsNull":false,"Value":false,"SubId":2}
],
"Layout":{"Offset":0,"Size":2,"IsLastPage":true,"IsFullPage":false}
}

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,
  "IsFullPage": true,
  "IsSubId": true,
  "SubId": 2,
  "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, if size <= 0 then all rows selected
IsFullPage - (optional) if true then always return non-empty last page of data.
IsSubId    - (optional) if true then select only single sub-value, default: all sub-values
SubId      - (optional) sub-value id to select if IsSubId is true
FilterById - (optional) conditions to filter dimension enum id's
OrderBy    - (optional) list of columns indexes (one based) to order by
```

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,"IsFullPage":false,"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,
  "IsFullPage": true
}
```

Result:

```
{"Layout":{"Offset":6,"Size":2,"IsFullPage":true,"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,"IsFullPage":false,"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": "isOldAge",
  "Offset": 0,
  "Size": 0,
  "IsSubId": true,
  "SubId": 2,
  "FilterById": [{
      "Name": "dim0",
      "Op": "IN",
      "EnumIds": [20, 40]
    }
  ],
  "OrderBy": [{
      "IndexOne": 2,
      "IsDesc": true
    }, {
      "IndexOne": 3,
      "IsDesc": true
    }
  ]
}
```

Result:


```
{
"Page":[
{"DimIds":[40],"IsNull":false,"Value":true,"SubId":2}
,{"DimIds":[20],"IsNull":false,"Value":false,"SubId":2}
],
"Layout":{"Offset":0,"Size":2,"IsLastPage":true,"IsFullPage":false}
}
```

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,
  "IsFullPage": true,
  "IsSubId": true,
  "SubId": 2,
  "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.
IsFullPage - (optional) if true then always return non-empty last page of data.
IsSubId    - (optional) if true then select only single sub-value, default: all sub-values
SubId      - (optional) sub-value id to select if IsSubId is true
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, ..., param_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,"IsFullPage":false,"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,
  "IsFullPage": true,
  "Filter":     [],
  "OrderBy":    []
}
```

Result:


```
{"Layout":{"Offset":6,"Size":2,"IsFullPage":true,"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,"IsFullPage":false,"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:

There are 3 filters applied to `ageSex` parameter:

- age group dimension: `dim0 IN ["20-30", "40+"]`
- AND sex dimension: `dim1 = "F"`
- AND parameter values: `param_value < 0.5`

JSON body:

```
{
  "Name": "ageSex",
  "Offset": 0,
  "Size": 100,
  "Filter": [{
      "Name": "dim0",
      "Op": "IN",
      "Values": ["20-30", "40+"]
    }, {
      "Name": "dim1",
      "Op": "=",
      "Values": ["F"]
    }, {
      "Name": "param_value",
      "Op": "<",
      "Values": ["0.5"]
    }
  ]
}
```

Result:


```
{
"Page":[
  {"Dims":["20-30","F"],"IsNull":false,"Value":0.4,"SubId":0}
],
"Layout":{"Offset":0,"Size":1,"IsLastPage":true,"IsFullPage":false}
}
```

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,
  "IsFullPage": true,
  "IsSubId": true,
  "SubId": 2,
  "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, if size <= 0 then all rows selected
IsFullPage - (optional) if true then always return non-empty last page of data.
IsSubId    - (optional) if true then select only single sub-value, default: all sub-values
SubId      - (optional) sub-value id to select if IsSubId is true
FilterById - (optional) conditions to filter dimension enum code(s)
OrderBy    - (optional) list of columns indexes (one based) to order by
```

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, ..., param_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,"IsFullPage":false,"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,
  "IsFullPage": true
}
```

Result:


```
{"Layout":{"Offset":6,"Size":2,"IsFullPage":true,"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,"IsFullPage":false,"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": "isOldAge",
  "Offset": 0,
  "Size": 0,
  "IsSubId": true,
  "SubId": 2,
  "FilterById": [{
      "Name": "dim0",
      "Op": "IN",
      "EnumIds": [20, 40]
    }
  ],
  "OrderBy": [{
      "IndexOne": 2,
      "IsDesc": true
    }, {
      "IndexOne": 3,
      "IsDesc": true
    }
  ]
}
```

Result:

{
"Page":[
{"DimIds":[40],"IsNull":false,"Value":true,"SubId":2}
,{"DimIds":[20],"IsNull":false,"Value":false,"SubId":2}
],
"Layout":{"Offset":0,"Size":2,"IsLastPage":true,"IsFullPage":false}
}

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,
  "IsFullPage": 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
    }
  ],
  "IsSubId": true,
  "SubId": 2,
  "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, if size <= 0 then all rows selected
IsFullPage - (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
IsSubId    - (optional) if true then select only single sub-value, default: all sub-values
SubId      - (optional) sub-value id to select if IsSubId is true
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
```

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:


```
{"Layout":{"Offset":0,"Size":36,"IsFullPage":false,"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":154.799999999999998,"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,
  "IsFullPage": true,
  "Filter":     [],
  "OrderBy":    []
}
```

Result:

{"Layout":{"Offset":32,"Size":4,"IsFullPage":true,"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,"IsFullPage":false,"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:

There are 3 filters applied to `salarySex` output table accumulator `acc1`, sub-value id = 2:

- salary level dimension: `dim0 IN ["L", "H"]`
- AND sex dimension: `dim1 BETWEEN ["F", "all"]`
- AND values of second sub-value of accumulator `acc1 > 5`

JSON body:

```
{
  "Name": "salarySex",
  "Offset": 0,
  "Size": 100,
  "Filter": [{
      "Name": "dim0",
      "Op": "IN",
      "Values": ["L", "H"]
    }, {
      "Name": "dim1",
      "Op": "BETWEEN",
      "Values": ["F", "all"]
    }, {
      "Name": "acc1",
      "Op": ">",
      "Values": ["5"]
    }
  ],
  "OrderBy": [{
      "IndexOne": 2,
      "IsDesc": true
    }, {
      "IndexOne": 3,
      "IsDesc": true
    }
  ],
  "IsSubId": true,
  "SubId": 2,
  "ValueName": "acc1",
  "IsAccum": true,
  "IsAllAccum": false
}
```


Result:


```
{
"Page":
[
{"Dims":["H","all"],"IsNull":false,"Value":804,"AccId":1,"SubId":2}
,{"Dims":["L","all"],"IsNull":false,"Value":802,"AccId":1,"SubId":2}
],
"Layout":{"Offset":0,"Size":2,"IsLastPage":true,"IsFullPage":false}
}
```

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,
  "IsFullPage": true,
  "IsSubId": true,
  "SubId": 2,
  "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, if size <= 0 then all rows selected
IsFullPage - (optional) if true then always return non-empty last page of data
IsSubId    - (optional) if true then select only single sub-value, default: all sub-values
SubId      - (optional) sub-value id to select if IsSubId is true
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
```

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,"IsFullPage":false,"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,
  "IsFullPage": true
}
```

Result:

```
{"Layout":{"Offset":32,"Size":4,"IsFullPage":true,"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,"IsFullPage":false,"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": 0,
  "IsSubId": true,
  "SubId": 2,
  "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": "acc1",
  "IsAccum": true,
  "IsAllAccum": true
}
```

Result:

{
"Page":[
{"DimIds":[300,800],"SubId":2,"IsNull":[false],"Value":[804]}
,{"DimIds":[300,1],"SubId":2,"IsNull":[false],"Value":[5]}
,{"DimIds":[100,800],"SubId":2,"IsNull":[false],"Value":[802]}
,{"DimIds":[100,1],"SubId":2,"IsNull":[false],"Value":[3]}
],
"Layout":{"Offset":0,"Size":4,"IsLastPage":true,"IsFullPage":false}
}

Read output table calculated values from model run

Read a "page" of output table calculated values from model run.

- Calculate one or more values from output table expressions or 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 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").
- Calculations are done either on output table expressions or aggregated accumulators (see example below).
- 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.ReadCalculteTableLayout structure from Go library. See also: db.ReadLayout structure from Go library.

```
// ReadCalculteTableLayout describe table read layout and additional measures to calculte.
type ReadCalculteTableLayout struct {
    ReadLayout                  // output table name, run id, page size, where filters and order by
    Calculation []CalculateTableLayout // additional measures to calculate
}

// CalculateLayout describes calculation to output table values.
type CalculateTableLayout struct {
    CalculateLayout      // expression to calculate and layout
    IsAggr          bool // if true then select output table accumulator else expression
}

// CalculateLayout describes calculation to parameters or output table values.
type CalculateLayout struct {
    Calculate string // expression to calculate, ex.: Expr0[base] - Expr0[variant]
    CalcId    int    // calculated expression id, calc_id column in csv,     ex.: 0, 12000, 24000
    Name      string // calculated expression name, calc_name column in csv, ex.: Expr0, AVG_Expr0, RATIO_Expro0
}

// ReadLayout describes source and size of data page to read input parameter, output table values or microdata.
//
// Row filters combined by AND and allow to select dimension or attribute items,
// it can be enum codes or enum id's, ex.: dim0 = 'CA' AND dim1 IN (2010, 2011, 2012)
type ReadLayout struct {
    Name           string           // parameter name, output table name or entity microdata name
    FromId         int              // run id or set id to select input parameter, output table values or microdata from
    ReadPageLayout                  // read page first row offset, size and last page flag
    Filter         []FilterColumn   // dimension or attribute filters, final WHERE does join all filters by AND
    FilterById     []FilterIdColumn // dimension or attribute filters by enum ids, final WHERE does join filters by AND
    OrderBy        []OrderByColumn  // order by columnns, if empty then dimension id ascending order is used
}
```

Method:

```
POST /api/model/:model/run/:run/table/calc
```

For example:

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/run/Default/table/calc -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/calc -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:

Example 1. Calculate two values using `modelOne` output table `salarySex`:

- ratio of expressions: `expr1 / expr2`
- standard error of `acc1` accumulator sub-values

```
{
"Name":      "salarySex",
"Calculation": [
  {
    "Calculate": "expr1 / expr2",
    "CalcId":    201,
    "Name":      "Expr1_div_expr2",
    "IsAggr": false
  },
  {
    "Calculate": "OM_SE(acc1)",
    "CalcId":    301,
    "Name":      "Se_of_acc1",
    "IsAggr": true
  }
]
}
```

Calcultion can be done over output table expressions if `IsAggr: false` or over accumulators if `IsAggr: true`. You cannot mix expressions and accumultors in the same calculation, it is mutually exclusive. Following aggregation functions avaliable for accumulators:

- `OM_AVG` mean of accumulators sub-values
- `OM_SUM` sum of accumulators sub-values
- `OM_COUNT` count of accumulators sub-values (excluding NULL's)
- `OM_COUNT_IF` count values matching condition
- `OM_MAX` maximum of accumulators sub-values
- `OM_MIN` minimum of accumulators sub-values
- `OM_VAR` variance of accumulators sub-values
- `OM_SD` standard deviation of accumulators sub-values
- `OM_SE` standard error of accumulators sub-values
- `OM_CV` coefficient of variation of accumulators sub-values

It is also possible to use parameter(s) in calculation, parameter must be a scalar of float or integer type. For example: `OM_COUNT_IF(acc1 > param.High)`, where `param.High` is a value of scalar parameter `High` in that model run.

For more details please see: Model Output Expressions

Example 2.:

- ratio of expressions: `expr1 / expr2`, adjusted by using parameter `StartingSeed` values
- standard error of `acc1` accumulator sub-values, adjusted by using parameter `StartingSeed` values
- read only first 100 rows: `Offset: 0, Size: 100`
- apply ORDER BY
- filter only rows where:
  - dimension Salary `dim0 IN ["L", "H"]`
  - and dimension Sex `dim1 BETWEEN "F" AND "all"`

- and value of `Expr1_div_expr2_adjusted < 11`
- and value of `Se_of_acc1_adjusted > 1023`

```
{
"Name": "salarySex",
"Calculation": [
{
 "Calculate": "expr1 / expr2 + param.StartingSeed / 100",
 "CalcId":    201,
 "Name":      "Expr1_div_expr2_adjusted",
 "IsAggr": false
},
{
 "Calculate": "OM_SE(acc1 - param.StartingSeed) + param.StartingSeed",
 "CalcId":    301,
 "Name":      "Se_of_acc1_adjusted",
 "IsAggr": true
}
],
"Offset": 0,
"Size": 100,
"IsFullPage": true,
"Filter": [{
 "Name": "dim0",
 "Op": "IN",
 "Values": ["L", "H"]
}, {
 "Name": "dim1",
 "Op": "BETWEEN",
 "Values": ["F", "all"]
}, {
 "Name": "Expr1_div_expr2_adjusted",
 "Op": "<",
 "Values": ["11"]
}, {
 "Name": "Se_of_acc1_adjusted",
 "Op": ">",
 "Values": ["1023"]
}
],
"OrderBy": [{
 "IndexOne": 2,
 "IsDesc": true
}, {
 "IndexOne": 3,
 "IsDesc": 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, if size <= 0 then all rows selected
IsFullPage - (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
```

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. First two columns are `run_id, calc_id`:

```
SELECT run_id, CalcId AS calc_id, dim0, dim1, ..., calc_value FROM .... 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",
  "Calculation": [
    {
      "Calculate": "expr1 / expr2",
      "CalcId":    201,
      "Name":      "Expr1_div_expr2",
      "IsAggr": false
    },
    {
      "Calculate": "OM_AVG(acc1)",
      "CalcId":    301,
      "Name":      "Se_of_acc1",
      "IsAggr": true
    }
  ]
}
```

Result:


```
{
    "Page": [{
            "Dims": ["L", "M"],
            "IsNull": false,
            "Value": 0,
            "CalcName": "Expr1_div_expr2",
            "RunDigest": "ca663651953bae94d0afdf71edba4c91"
        }, {
            "Dims": ["L", "F"],
            "IsNull": false,
            "Value": 0.01639344262295082,
            "CalcName": "Expr1_div_expr2",
            "RunDigest": "ca663651953bae94d0afdf71edba4c91"
        }, {
            "Dims": ["L", "all"],
            "IsNull": false,
            "Value": 0.9987515605493134,
            "CalcName": "Expr1_div_expr2",
            "RunDigest": "ca663651953bae94d0afdf71edba4c91"
        }, {
            "Dims": ["M", "M"],
            "IsNull": false,
            "Value": 0.019011406844106467,
            "CalcName": "Expr1_div_expr2",
            "RunDigest": "ca663651953bae94d0afdf71edba4c91"
        }, {
            "Dims": ["M", "F"],
            "IsNull": false,
            "Value": 0.03125,
            "CalcName": "Expr1_div_expr2",
            "RunDigest": "ca663651953bae94d0afdf71edba4c91"
        }, {
            "Dims": ["M", "all"],
            "IsNull": false,
            "Value": 0.9975093399750934,
            "CalcName": "Expr1_div_expr2",
            "RunDigest": "ca663651953bae94d0afdf71edba4c91"
        }, {
            "Dims": ["H", "M"],
            "IsNull": false,
            "Value": 0.036231884057971016,
            "CalcName": "Expr1_div_expr2",
            "RunDigest": "ca663651953bae94d0afdf71edba4c91"
        }, {
            "Dims": ["H", "F"],
            "IsNull": false,
            "Value": 0.04477611940298507,
            "CalcName": "Expr1_div_expr2",
```

```
            "RunDigest": "ca663651953bae94d0afdf71edba4c91"
        }, {
            "Dims": ["H", "all"],
            "IsNull": false,
            "Value": 0.9962732919254659,
            "CalcName": "Expr1_div_expr2",
            "RunDigest": "ca663651953bae94d0afdf71edba4c91"
        }, {
            "Dims": ["L", "M"],
            "IsNull": false,
            "Value": 0,
            "CalcName": "Se_of_acc1",
            "RunDigest": "ca663651953bae94d0afdf71edba4c91"
        }, {
            "Dims": ["L", "F"],
            "IsNull": false,
            "Value": 1,
            "CalcName": "Se_of_acc1",
            "RunDigest": "ca663651953bae94d0afdf71edba4c91"
        }, {
            "Dims": ["L", "all"],
            "IsNull": false,
            "Value": 800,
            "CalcName": "Se_of_acc1",
            "RunDigest": "ca663651953bae94d0afdf71edba4c91"
        }, {
            "Dims": ["M", "M"],
            "IsNull": false,
            "Value": 1,
            "CalcName": "Se_of_acc1",
            "RunDigest": "ca663651953bae94d0afdf71edba4c91"
        }, {
            "Dims": ["M", "F"],
            "IsNull": false,
            "Value": 2,
            "CalcName": "Se_of_acc1",
            "RunDigest": "ca663651953bae94d0afdf71edba4c91"
        }, {
            "Dims": ["M", "all"],
            "IsNull": false,
            "Value": 801,
            "CalcName": "Se_of_acc1",
            "RunDigest": "ca663651953bae94d0afdf71edba4c91"
        }, {
            "Dims": ["H", "M"],
            "IsNull": false,
            "Value": 2,
            "CalcName": "Se_of_acc1",
            "RunDigest": "ca663651953bae94d0afdf71edba4c91"
        }, {
            "Dims": ["H", "F"],
            "IsNull": false,
            "Value": 3,
            "CalcName": "Se_of_acc1",
            "RunDigest": "ca663651953bae94d0afdf71edba4c91"
        }, {
            "Dims": ["H", "all"],
            "IsNull": false,
            "Value": 802,
            "CalcName": "Se_of_acc1",
            "RunDigest": "ca663651953bae94d0afdf71edba4c91"
        }
    ],
    "Layout": {
        "Offset": 0,
        "Size": 18,
        "IsLastPage": true,
        "IsFullPage": false
    }
}
```


Example 2:

JSON body:

```
{
  "Name": "salarySex",
  "Calculation": [
    {
      "Calculate": "expr1 / expr2 + param.StartingSeed / 100",
      "CalcId":    201,
      "Name":      "Expr1_div_expr2_adjusted",
      "IsAggr": false
    },
    {
      "Calculate": "OM_SE(acc1 - param.StartingSeed) + param.StartingSeed",
      "CalcId":    301,
      "Name":      "Se_of_acc1_adjusted",
      "IsAggr": true
    }
  ],
  "Offset": 0,
  "Size": 100,
  "IsFullPage": true,
  "Filter": [{
      "Name": "dim0",
      "Op": "IN",
      "Values": ["L", "H"]
    }, {
      "Name": "dim1",
      "Op": "BETWEEN",
      "Values": ["F", "all"]
    }, {
      "Name": "Expr1_div_expr2_adjusted",
      "Op": "<",
      "Values": ["11"]
    }, {
      "Name": "Se_of_acc1_adjusted",
      "Op": ">",
      "Values": ["1023"]
    }
  ],
  "OrderBy": [{
      "IndexOne": 2,
      "IsDesc": true
    }, {
      "IndexOne": 3,
      "IsDesc": true
    }
  ]
}
```


Result:

```
{
  "Page": [{
      "Dims": ["H", "F"],
      "IsNull": false,
      "Value": 1023.6454972243679,
      "CalcName": "Se_of_acc1_adjusted",
      "RunDigest": "cdc58c932f4a5d3a90b2932f94dc41b7"
    }, {
      "Dims": ["H", "all"],
      "IsNull": false,
      "Value": 1023.6454972243679,
      "CalcName": "Se_of_acc1_adjusted",
      "RunDigest": "cdc58c932f4a5d3a90b2932f94dc41b7"
    }, {
      "Dims": ["L", "F"],
      "IsNull": false,
      "Value": 1023.6454972243679,
      "CalcName": "Se_of_acc1_adjusted",
      "RunDigest": "cdc58c932f4a5d3a90b2932f94dc41b7"
    }, {
      "Dims": ["L", "all"],
      "IsNull": false,
      "Value": 1023.6454972243679,
      "CalcName": "Se_of_acc1_adjusted",
      "RunDigest": "cdc58c932f4a5d3a90b2932f94dc41b7"
    }, {
      "Dims": ["H", "F"],
      "IsNull": false,
      "Value": 10.492773722627737,
      "CalcName": "Expr1_div_expr2_adjusted",
      "RunDigest": "cdc58c932f4a5d3a90b2932f94dc41b7"
    }, {
      "Dims": ["L", "F"],
      "IsNull": false,
      "Value": 10.39,
      "CalcName": "Expr1_div_expr2_adjusted",
      "RunDigest": "cdc58c932f4a5d3a90b2932f94dc41b7"
    }
  ],
  "Layout": {
    "Offset": 0,
    "Size": 6,
    "IsLastPage": true,
    "IsFullPage": true
  }
}
```

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

Read a "page" of output table calculated values from model run.

- Calculate one or more values from output table expressions or 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 below start row number returned.
- Dimension(s) returned as enum id, not enum codes.
- Calculations are done either on output table expressions or aggregated accumulators (see example below).
- 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.ReadCalculteTableLayout structure from Go library. See also: db.ReadLayout structure from Go library.

```
// ReadCalculteTableLayout describe table read layout and additional measures to calculte.
type ReadCalculteTableLayout struct {
    ReadLayout                  // output table name, run id, page size, where filters and order by
    Calculation []CalculateTableLayout // additional measures to calculate
}

// CalculateLayout describes calculation to output table values.
type CalculateTableLayout struct {
    CalculateLayout      // expression to calculate and layout
    IsAggr          bool // if true then select output table accumulator else expression
}

// CalculateLayout describes calculation to parameters or output table values.
type CalculateLayout struct {
    Calculate string // expression to calculate, ex.: Expr0[base] - Expr0[variant]
    CalcId    int    // calculated expression id, calc_id column in csv,     ex.: 0, 12000, 24000
    Name      string // calculated expression name, calc_name column in csv, ex.: Expr0, AVG_Expr0, RATIO_Expro0
}

// ReadLayout describes source and size of data page to read input parameter, output table values or microdata.
//
// Row filters combined by AND and allow to select dimension or attribute items,
// it can be enum codes or enum id's, ex.: dim0 = 'CA' AND dim1 IN (2010, 2011, 2012)
type ReadLayout struct {
    Name           string           // parameter name, output table name or entity microdata name
    FromId         int              // run id or set id to select input parameter, output table values or microdata from
    ReadPageLayout                  // read page first row offset, size and last page flag
    Filter         []FilterColumn   // dimension or attribute filters, final WHERE does join all filters by AND
    FilterById     []FilterIdColumn // dimension or attribute filters by enum ids, final WHERE does join filters by AND
    OrderBy        []OrderByColumn  // order by columnns, if empty then dimension id ascending order is used
}
```

Method:

```
POST /api/model/:model/run/:run/table/calc-id
```

For example:

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/run/Default/table/calc-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/calc-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:

Example 1. Calculate two values using `modelOne` output table `salarySex`:

- ratio of expressions: `expr1 / expr2`
- standard error of `acc1` accumulator sub-values

```
{
"Name":      "salarySex",
"Calculation": [
  {
    "Calculate": "expr1 / expr2",
    "CalcId":    201,
    "Name":      "Expr1_div_expr2",
    "IsAggr": false
  },
  {
    "Calculate": "OM_SE(acc1)",
    "CalcId":    301,
    "Name":      "Se_of_acc1",
    "IsAggr": true
  }
]
}
```

  Calcultion can be done over output table expressions if `IsAggr: false` or over accumulators if `IsAggr: true`. You cannot mix expressions and accumultors in the same calculation, it is mutually exclusive. Following aggregation functions avaliable for accumulators:

- `OM_AVG` mean of accumulators sub-values
- `OM_SUM` sum of accumulators sub-values
- `OM_COUNT` count of accumulators sub-values (excluding NULL's)
- `OM_COUNT_IF` count values matching condition
- `OM_MAX` maximum of accumulators sub-values
- `OM_MIN` minimum of accumulators sub-values
- `OM_VAR` variance of accumulators sub-values
- `OM_SD` standard deviation of accumulators sub-values
- `OM_SE` standard error of accumulators sub-values
- `OM_CV` coefficient of variation of accumulators sub-values

It is also possible to use parameter(s) in calculation, parameter must be a scalar of float or integer type. For example: `OM_COUNT_IF(acc1 > param.High)`, where `param.High` is a value of scalar parameter `High` in that model run.

For more details please see: Model Output Expressions

Example 2:

- ratio of expressions: `expr1 / expr2`, adjusted by using parameter `StartingSeed` values
- standard error of `acc1` accumulator sub-values, adjusted by using parameter `StartingSeed` values
- read only first 100 rows: `Offset: 0, Size: 100`
- apply WHERE filters and ORDER BY

```
{
"Name": "salarySex",
"Calculation": [
  {
    "Calculate": "expr1 / expr2 + param.StartingSeed / 100",
    "CalcId":    201,
    "Name":      "Expr1_div_expr2_adjusted",
    "IsAggr": false
  },
  {
    "Calculate": "OM_SE(acc1 - param.StartingSeed) + param.StartingSeed",
    "CalcId":    301,
    "Name":      "Se_of_acc1_adjusted",
    "IsAggr": true
  }
],
"Offset": 0,
"Size": 100,
"IsFullPage": true,
"IsSubId": true,
"SubId": 2,
"FilterById": [{
    "Name": "AgeGroup",
    "Op": "IN",
    "EnumIds": [100, 300]
  }, {
    "Name": "Province",
    "Op": "BETWEEN",
    "EnumIds": [1, 800]
  }
],
"OrderBy": [{
    "IndexOne": 2,
    "IsDesc": true
  }, {
    "IndexOne": 3,
    "IsDesc": 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, if size <= 0 then all rows selected
IsFullPage - (optional) if true then always return non-empty last page of data
IsSubId    - (optional) if true then select only single sub-value, default: all sub-values
SubId      - (optional) sub-value id to select if IsSubId is true
FilterById - (optional) conditions to filter dimension enum id's
OrderBy    - (optional) list of columns indexes (one based) to order by
```

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 First two columns are `run_id, calc_id`:

```
SELECT run_id, CalcId AS calc_id, dim0, dim1, ..., calc_value FROM .... ORDER BY 1, 2,...
```

Example 1:

JSON body:

```
{
  "Name":      "salarySex",
  "Calculation": [
    {
      "Calculate": "expr1 / expr2",
      "CalcId":    201,
      "Name":      "Expr1_div_expr2",
      "IsAggr": false
    },
    {
      "Calculate": "OM_AVG(acc1)",
      "CalcId":    301,
      "Name":      "Se_of_acc1",
      "IsAggr": true
    }
  ]
}
```


Result:


```
> POST /api/model/modelOne/run/Default/table/calc-id HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/8.0.1
> Accept: */*
> Content-Type: application/json
> Content-Length: 296
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Date: Thu, 26 Oct 2023 02:52:01 GMT
< Content-Length: 1489
<
{
    "Page": [{
            "DimIds": [100, 0],
            "IsNull": false,
            "Value": 0,
            "CalcId": 201,
            "RunId": 201
        }, {
            "DimIds": [100, 1],
            "IsNull": false,
            "Value": 0.01639344262295082,
            "CalcId": 201,
            "RunId": 201
        }, {
            "DimIds": [100, 800],
            "IsNull": false,
            "Value": 0.9987515605493134,
            "CalcId": 201,
            "RunId": 201
        }, {
            "DimIds": [200, 0],
            "IsNull": false,
            "Value": 0.019011406844106467,
            "CalcId": 201,
            "RunId": 201
        }, {
            "DimIds": [200, 1],
            "IsNull": false,
            "Value": 0.03125,
            "CalcId": 201,
            "RunId": 201
        }, {
            "DimIds": [200, 800],
            "IsNull": false,
            "Value": 0.9975093399750934,
            "CalcId": 201,
            "RunId": 201
        }, {
            "DimIds": [300, 0],
            "IsNull": false,
            "Value": 0.036231884057971016,
            "CalcId": 201,
            "RunId": 201
        }, {
            "DimIds": [300, 1],
            "IsNull": false,
            "Value": 0.04477611940298507,
            "CalcId": 201,
            "RunId": 201
        }, {
            "DimIds": [300, 800],
            "IsNull": false,
            "Value": 0.9962732919254659,
```

```
            "CalcId": 201,
            "RunId": 201
        }, {
            "DimIds": [100, 0],
            "IsNull": false,
            "Value": 0,
            "CalcId": 301,
            "RunId": 201
        }, {
            "DimIds": [100, 1],
            "IsNull": false,
            "Value": 1,
            "CalcId": 301,
            "RunId": 201
        }, {
            "DimIds": [100, 800],
            "IsNull": false,
            "Value": 800,
            "CalcId": 301,
            "RunId": 201
        }, {
            "DimIds": [200, 0],
            "IsNull": false,
            "Value": 1,
            "CalcId": 301,
            "RunId": 201
        }, {
            "DimIds": [200, 1],
            "IsNull": false,
            "Value": 2,
            "CalcId": 301,
            "RunId": 201
        }, {
            "DimIds": [200, 800],
            "IsNull": false,
            "Value": 801,
            "CalcId": 301,
            "RunId": 201
        }, {
            "DimIds": [300, 0],
            "IsNull": false,
            "Value": 2,
            "CalcId": 301,
            "RunId": 201
        }, {
            "DimIds": [300, 1],
            "IsNull": false,
            "Value": 3,
            "CalcId": 301,
            "RunId": 201
        }, {
            "DimIds": [300, 800],
            "IsNull": false,
            "Value": 802,
            "CalcId": 301,
            "RunId": 201
        }
    ],
    "Layout": {
        "Offset": 0,
        "Size": 18,
        "IsLastPage": true,
        "IsFullPage": false
    }
}
```

Read output table values and compare model runs

Read a "page" of output table values and compare model runs.

- Compare output table expressions between multiple model runs.
- Comparison typically is a calculation between `[base]` and `[variant]` model runs, for example: `Expr0[variant] / Expr0[base]`.
- It is also posiible to include calculation results for each single run, for example: `Expr0` or `100 * Expr0 / Expr1`.
- 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").
- 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.ReadCompareTableLayout structure from Go library. See also: db.ReadLayout structure from Go library.

```
// ReadCompareTableLayout to compare output table runs with base run using multiple comparison expressions and/or calculation measures.
type ReadCompareTableLayout struct {
    ReadCalculteTableLayout          // output table, base run and comparison expressions or calculations
    Runs                    []string // runs to compare: list of digest, stamp or name
}

// ReadCalculteTableLayout describe table read layout and additional measures to calculte.
type ReadCalculteTableLayout struct {
    ReadLayout                      // output table name, run id, page size, where filters and order by
    Calculation []CalculateTableLayout // additional measures to calculate
}

// CalculateLayout describes calculation to output table values.
type CalculateTableLayout struct {
    CalculateLayout      // expression to calculate and layout
    IsAggr          bool // if true then select output table accumulator else expression
}

// CalculateLayout describes calculation to parameters or output table values.
type CalculateLayout struct {
    Calculate string // expression to calculate, ex.: Expr0[base] - Expr0[variant]
    CalcId    int    // calculated expression id, calc_id column in csv,     ex.: 0, 12000, 24000
    Name      string // calculated expression name, calc_name column in csv, ex.: Expr0, AVG_Expr0, RATIO_Expro0
}

// ReadLayout describes source and size of data page to read input parameter, output table values or microdata.
//
// Row filters combined by AND and allow to select dimension or attribute items,
// it can be enum codes or enum id's, ex.: dim0 = 'CA' AND dim1 IN (2010, 2011, 2012)
type ReadLayout struct {
    Name           string           // parameter name, output table name or entity microdata name
    FromId         int              // run id or set id to select input parameter, output table values or microdata from
    ReadPageLayout                  // read page first row offset, size and last page flag
    Filter         []FilterColumn   // dimension or attribute filters, final WHERE does join all filters by AND
    FilterById     []FilterIdColumn // dimension or attribute filters by enum ids, final WHERE does join filters by AND
    OrderBy        []OrderByColumn  // order by columnns, if empty then dimension id ascending order is used
}
```

Method:

```
POST /api/model/:model/run/:run/table/compare
```

For example:

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/run/Default/table/compare -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/compare -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:

Example 1. For `modelOne` output table `salarySex`:

- read `expr0` values from `[base]` run and from `[variant]` runs: "Default-4", "First Task Run_modelOne_other"
- calculate difference of `expr0` values between `[variant]` runs and `[base]`

```
{
"Name":      "salarySex",
"Calculation": [
  {
    "Calculate": "expr0",
    "CalcId":    0,
    "Name":      "expr0",
    "IsAggr": false
  },
  {
    "Calculate": "expr0[variant] - expr0[base]",
    "CalcId":    12000,
    "Name":      "Diff_of_expr0",
    "IsAggr": false
  }
],
"Runs": [
  "Default-4",
  "First Task Run_modelOne_other"
]
}
```

  Calculation must be done over output table expressions and can NOT include table accumulators. Output table expression can be from `[base]` or from `[variant]` model run, e.g.: `expr0[variant] - expr0[base]`. It is also possible to use to do a calcultion for each single (not between two runs), e.g.: `expr0`, or `100 * expr0 / expr1`. Calcultion must be a comparison formula between two runs `[base]` and `[variant]` or done on single run. You cannot mix comparison and sincle run calcultion, it is mutually exclusive, for exampel: thisi is an error: `expr1 + expr0[variant] - expr0[base]`. Calculation can include expression names, `+ - * /` operators and following functions:

- `SQRT` square root
- `ABS` absolute value
- `OM_IF` equivalent of `if .... then .... else ....`
- `OM_DIV_BY` wrap denominator

It is also possible to use parameter(s) in calculation, parameter must be a scalar of float or integer type. For example: `(Expr1[variant] - Expr1[base]) * param.Scale[base]`, where `param.Scale` is a value of scalar parameter `Scale` in `[base]` model run.

For more details please see: Model Output Expressions

Example 2.:

- read `expr0` values from `[base]` run and from `[variant]` runs: "Default-4", "First Task Run_modelOne_other"
- calculate difference of `expr0` values between `[variant]` runs and `[base]`, adjusted by using parameter `StartingSeed` value from `base` run
- read only first 100 rows: `Offset: 0, Size: 100`
- apply ORDER BY
- filter only rows where:
  - dimension Salary `dim0 IN ["L", "H"]`

- and dimension Sex `dim1 BETWEEN "F" AND "all"`
- and value of `expr0 > 200`
- and value of `Diff_of_expr0_adjusted > 300`

```
{
"Name":      "salarySex",
"Calculation": [
{
 "Calculate": "expr0",
 "CalcId":    0,
 "Name":      "expr0",
 "IsAggr": false
},
{
 "Calculate": "(expr0[variant] - expr0[base]) + param.StartingSeed[base] / 100",
 "CalcId":    12000,
 "Name":      "Diff_of_expr0_adjusted",
 "IsAggr": false
}
],
"Runs": [
"Default-4",
"First Task Run_modelOne_other"
],
"Offset": 0,
"Size": 100,
"IsFullPage": true,
"Filter": [{
 "Name": "dim0",
 "Op": "IN",
 "Values": ["L", "H"]
}, {
 "Name": "dim1",
 "Op": "BETWEEN",
 "Values": ["F", "all"]
}, {
 "Name": "expr0",
 "Op": ">",
 "Values": ["200"]
}, {
 "Name": "Diff_of_expr0_adjusted",
 "Op": ">",
 "Values": ["300"]
}
],
"OrderBy": [{
 "IndexOne": 2,
 "IsDesc": true
}, {
 "IndexOne": 3,
 "IsDesc": 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, if size <= 0 then all rows selected
IsFullPage - (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
```

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. First two columns are `run_id, calc_id`:

```
SELECT run_id, CalcId AS calc_id, dim0, dim1, ..., calc_value FROM .... 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:

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/run/Default/table/compare -d @test.json
```

JSON body (test.json):

```
{
  "Name":      "salarySex",
  "Calculation": [
    {
      "Calculate": "expr0",
      "CalcId":    0,
      "Name":      "expr0",
      "IsAggr": false
    },
    {
      "Calculate": "expr0[variant] - expr0[base]",
      "CalcId":    12000,
      "Name":      "Diff_of_expr0",
      "IsAggr": false
    }
  ],
  "Runs": [
    "Default-4",
    "First Task Run_modelOne_other"
  ]
}
```

Result:


```
{
    "Page": [{
            "Dims": ["L", "M"],
            "IsNull": false,
            "Value": 50,
            "CalcName": "expr0",
            "RunDigest": "ca663651953bae94d0afdf71edba4c91"
        }, {
            "Dims": ["L", "F"],
            "IsNull": false,
            "Value": 60,
            "CalcName": "expr0",
            "RunDigest": "ca663651953bae94d0afdf71edba4c91"
        }, {
            "Dims": ["L", "all"],
            "IsNull": false,
            "Value": 1,
            "CalcName": "expr0",
            "RunDigest": "ca663651953bae94d0afdf71edba4c91"
        }, {
            "Dims": ["M", "M"],
            "IsNull": false,
            "Value": 51.599999999999994,
            "CalcName": "expr0",
            "RunDigest": "ca663651953bae94d0afdf71edba4c91"
        }, {
            "Dims": ["M", "F"],
            "IsNull": false,
            "Value": 62,
            "CalcName": "expr0",
            "RunDigest": "ca663651953bae94d0afdf71edba4c91"
        }, {
            "Dims": ["M", "all"],
```

```
        "IsNull": false,
        "Value": 2,
        "CalcName": "expr0",
        "RunDigest": "ca663651953bae94d0afdf71edba4c91"
    }, {
        "Dims": ["H", "M"],
        "IsNull": false,
        "Value": 53.2,
        "CalcName": "expr0",
        "RunDigest": "ca663651953bae94d0afdf71edba4c91"
    }, {
        "Dims": ["H", "F"],
        "IsNull": false,
        "Value": 64,
        "CalcName": "expr0",
        "RunDigest": "ca663651953bae94d0afdf71edba4c91"
    }, {
        "Dims": ["H", "all"],
        "IsNull": false,
        "Value": 3,
        "CalcName": "expr0",
        "RunDigest": "ca663651953bae94d0afdf71edba4c91"
    }, {
        "Dims": ["L", "M"],
        "IsNull": false,
        "Value": 50,
        "CalcName": "expr0",
        "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
    }, {
        "Dims": ["L", "F"],
        "IsNull": false,
        "Value": 60,
        "CalcName": "expr0",
        "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
    }, {
        "Dims": ["L", "all"],
        "IsNull": false,
        "Value": 1201,
        "CalcName": "expr0",
        "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
    }, {
        "Dims": ["M", "M"],
        "IsNull": false,
        "Value": 51.599999999999994,
        "CalcName": "expr0",
        "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
    }, {
        "Dims": ["M", "F"],
        "IsNull": false,
        "Value": 62,
        "CalcName": "expr0",
        "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
    }, {
        "Dims": ["M", "all"],
        "IsNull": false,
        "Value": 1202,
        "CalcName": "expr0",
        "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
    }, {
        "Dims": ["H", "M"],
        "IsNull": false,
        "Value": 53.2,
        "CalcName": "expr0",
        "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
    }, {
        "Dims": ["H", "F"],
        "IsNull": false,
        "Value": 64,
        "CalcName": "expr0",
        "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
    }, {
        "Dims": ["H", "all"],
        "IsNull": false,
        "Value": 1203,
        "CalcName": "expr0",
        "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
    }, {
        "Dims": ["L", "M"],
        "IsNull": false,
        "Value": 0,
        "CalcName": "Diff_of_expr0",
        "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
    }, {
        "Dims": ["L", "F"],
        "IsNull": false,
        "Value": 0,
        "CalcName": "Diff_of_expr0",
        "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
```

```
    }, {
        "Dims": ["L", "all"],
        "IsNull": false,
        "Value": 1200,
        "CalcName": "Diff_of_expr0",
        "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
    }, {
        "Dims": ["M", "M"],
        "IsNull": false,
        "Value": 0,
        "CalcName": "Diff_of_expr0",
        "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
    }, {
        "Dims": ["M", "F"],
        "IsNull": false,
        "Value": 0,
        "CalcName": "Diff_of_expr0",
        "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
    }, {
        "Dims": ["M", "all"],
        "IsNull": false,
        "Value": 1200,
        "CalcName": "Diff_of_expr0",
        "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
    }, {
        "Dims": ["H", "M"],
        "IsNull": false,
        "Value": 0,
        "CalcName": "Diff_of_expr0",
        "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
    }, {
        "Dims": ["H", "F"],
        "IsNull": false,
        "Value": 0,
        "CalcName": "Diff_of_expr0",
        "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
    }, {
        "Dims": ["H", "all"],
        "IsNull": false,
        "Value": 1200,
        "CalcName": "Diff_of_expr0",
        "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
    }, {
        "Dims": ["L", "M"],
        "IsNull": false,
        "Value": 225.6,
        "CalcName": "expr0",
        "RunDigest": "9a1121a0392aa3eddd0932d269838e2d"
    }, {
        "Dims": ["L", "F"],
        "IsNull": false,
        "Value": 272,
        "CalcName": "expr0",
        "RunDigest": "9a1121a0392aa3eddd0932d269838e2d"
    }, {
        "Dims": ["L", "all"],
        "IsNull": false,
        "Value": 1,
        "CalcName": "expr0",
        "RunDigest": "9a1121a0392aa3eddd0932d269838e2d"
    }, {
        "Dims": ["M", "M"],
        "IsNull": false,
        "Value": 232,
        "CalcName": "expr0",
        "RunDigest": "9a1121a0392aa3eddd0932d269838e2d"
    }, {
        "Dims": ["M", "F"],
        "IsNull": false,
        "Value": 280,
        "CalcName": "expr0",
        "RunDigest": "9a1121a0392aa3eddd0932d269838e2d"
    }, {
        "Dims": ["M", "all"],
        "IsNull": false,
        "Value": 2,
        "CalcName": "expr0",
        "RunDigest": "9a1121a0392aa3eddd0932d269838e2d"
    }, {
        "Dims": ["H", "M"],
        "IsNull": false,
        "Value": 238.39999999999998,
        "CalcName": "expr0",
        "RunDigest": "9a1121a0392aa3eddd0932d269838e2d"
    }, {
        "Dims": ["H", "F"],
        "IsNull": false,
        "Value": 288
```

```
            "Value": 288,
            "CalcName": "expr0",
            "RunDigest": "9a1121a0392aa3eddd0932d269838e2d"
        }, {
            "Dims": ["H", "all"],
            "IsNull": false,
            "Value": 3,
            "CalcName": "expr0",
            "RunDigest": "9a1121a0392aa3eddd0932d269838e2d"
        }, {
            "Dims": ["L", "M"],
            "IsNull": false,
            "Value": 175.6,
            "CalcName": "Diff_of_expr0",
            "RunDigest": "9a1121a0392aa3eddd0932d269838e2d"
        }, {
            "Dims": ["L", "F"],
            "IsNull": false,
            "Value": 212,
            "CalcName": "Diff_of_expr0",
            "RunDigest": "9a1121a0392aa3eddd0932d269838e2d"
        }, {
            "Dims": ["L", "all"],
            "IsNull": false,
            "Value": 0,
            "CalcName": "Diff_of_expr0",
            "RunDigest": "9a1121a0392aa3eddd0932d269838e2d"
        }, {
            "Dims": ["M", "M"],
            "IsNull": false,
            "Value": 180.4,
            "CalcName": "Diff_of_expr0",
            "RunDigest": "9a1121a0392aa3eddd0932d269838e2d"
        }, {
            "Dims": ["M", "F"],
            "IsNull": false,
            "Value": 218,
            "CalcName": "Diff_of_expr0",
            "RunDigest": "9a1121a0392aa3eddd0932d269838e2d"
        }, {
            "Dims": ["M", "all"],
            "IsNull": false,
            "Value": 0,
            "CalcName": "Diff_of_expr0",
            "RunDigest": "9a1121a0392aa3eddd0932d269838e2d"
        }, {
            "Dims": ["H", "M"],
            "IsNull": false,
            "Value": 185.2,
            "CalcName": "Diff_of_expr0",
            "RunDigest": "9a1121a0392aa3eddd0932d269838e2d"
        }, {
            "Dims": ["H", "F"],
            "IsNull": false,
            "Value": 224,
            "CalcName": "Diff_of_expr0",
            "RunDigest": "9a1121a0392aa3eddd0932d269838e2d"
        }, {
            "Dims": ["H", "all"],
            "IsNull": false,
            "Value": 0,
            "CalcName": "Diff_of_expr0",
            "RunDigest": "9a1121a0392aa3eddd0932d269838e2d"
        }
    ],
    "Layout": {
        "Offset": 0,
        "Size": 45,
        "IsLastPage": true,
        "IsFullPage": false
    }
}
```


Example 1:

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/run/Default/table/compare -d @test2.json
```

JSON body (test2.json):

```
{
  "Name":      "salarySex",
  "Calculation": [
    {
      "Calculate": "expr0",
      "CalcId":    0,
      "Name":      "expr0",
      "IsAggr": false
    },
    {
      "Calculate": "(expr0[variant] - expr0[base]) + param.StartingSeed[base] / 100",
      "CalcId":    12000,
      "Name":      "Diff_of_expr0_adjusted",
      "IsAggr": false
    }
  ],
  "Runs": [
    "Default-4",
    "First Task Run_modelOne_other"
  ],
  "Offset": 0,
  "Size": 100,
  "IsFullPage": true,
  "Filter": [{
      "Name": "dim0",
      "Op": "IN",
      "Values": ["L", "H"]
    }, {
      "Name": "dim1",
      "Op": "BETWEEN",
      "Values": ["F", "all"]
    }, {
      "Name": "expr0",
      "Op": ">",
      "Values": ["200"]
    }, {
      "Name": "Diff_of_expr0_adjusted",
      "Op": ">",
      "Values": ["300"]
    }
  ],
  "OrderBy": [{
      "IndexOne": 2,
      "IsDesc": true
    }, {
      "IndexOne": 3,
      "IsDesc": true
    }
  ]
}
```


Result:

```
{
  "Page": [{
      "Dims": ["H", "all"],
      "IsNull": false,
      "Value": 1281.91,
      "CalcName": "Diff_of_expr0_adjusted",
      "RunDigest": "cdc58c932f4a5d3a90b2932f94dc41b7"
    }, {
      "Dims": ["H", "F"],
      "IsNull": false,
      "Value": 305.90999999999997,
      "CalcName": "Diff_of_expr0_adjusted",
      "RunDigest": "4f141ef2cb9e16bf6b753d21f7dfd8b9"
    }, {
      "Dims": ["L", "all"],
      "IsNull": false,
      "Value": 1281.91,
      "CalcName": "Diff_of_expr0_adjusted",
      "RunDigest": "cdc58c932f4a5d3a90b2932f94dc41b7"
    }, {
      "Dims": ["H", "all"],
      "IsNull": false,
      "Value": 1203,
      "CalcName": "expr0",
      "RunDigest": "cdc58c932f4a5d3a90b2932f94dc41b7"
    }, {
      "Dims": ["H", "F"],
      "IsNull": false,
      "Value": 288,
      "CalcName": "expr0",
      "RunDigest": "4f141ef2cb9e16bf6b753d21f7dfd8b9"
    }, {
      "Dims": ["L", "all"],
      "IsNull": false,
      "Value": 1201,
      "CalcName": "expr0",
      "RunDigest": "cdc58c932f4a5d3a90b2932f94dc41b7"
    }, {
      "Dims": ["L", "F"],
      "IsNull": false,
      "Value": 272,
      "CalcName": "expr0",
      "RunDigest": "4f141ef2cb9e16bf6b753d21f7dfd8b9"
    }
  ],
  "Layout": {
    "Offset": 0,
    "Size": 7,
    "IsLastPage": true,
    "IsFullPage": true
  }
}
```

Read output table values and compare model runs (enun id's)

Read a "page" of output table values and compare model runs.

- Compare output table expressions between multiple model runs.
- Comparison typically is a calculation between `[base]` and `[variant]` model runs, for example: `Expr0[variant] / Expr0[base]`.
- It is also posiible to include calculation results for each single run, for example: `Expr0` or `100 * Expr0 / Expr1`.
- 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.
- 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.ReadCompareTableLayout structure from Go library. See also: db.ReadLayout structure from Go library.

```
// ReadCompareTableLayout to compare output table runs with base run using multiple comparison expressions and/or calculation measures.
type ReadCompareTableLayout struct {
    ReadCalculteTableLayout          // output table, base run and comparison expressions or calculations
    Runs                    []string // runs to compare: list of digest, stamp or name
}

// ReadCalculteTableLayout describe table read layout and additional measures to calculte.
type ReadCalculteTableLayout struct {
    ReadLayout                      // output table name, run id, page size, where filters and order by
    Calculation []CalculateTableLayout // additional measures to calculate
}

// CalculateLayout describes calculation to output table values.
type CalculateTableLayout struct {
    CalculateLayout      // expression to calculate and layout
    IsAggr          bool // if true then select output table accumulator else expression
}

// CalculateLayout describes calculation to parameters or output table values.
type CalculateLayout struct {
    Calculate string // expression to calculate, ex.: Expr0[base] - Expr0[variant]
    CalcId    int    // calculated expression id, calc_id column in csv,     ex.: 0, 12000, 24000
    Name      string // calculated expression name, calc_name column in csv, ex.: Expr0, AVG_Expr0, RATIO_Expro0
}

// ReadLayout describes source and size of data page to read input parameter, output table values or microdata.
//
// Row filters combined by AND and allow to select dimension or attribute items,
// it can be enum codes or enum id's, ex.: dim0 = 'CA' AND dim1 IN (2010, 2011, 2012)
type ReadLayout struct {
    Name          string           // parameter name, output table name or entity microdata name
    FromId        int              // run id or set id to select input parameter, output table values or microdata from
    ReadPageLayout                  // read page first row offset, size and last page flag
    Filter        []FilterColumn   // dimension or attribute filters, final WHERE does join all filters by AND
    FilterById    []FilterIdColumn // dimension or attribute filters by enum ids, final WHERE does join filters by AND
    OrderBy       []OrderByColumn  // order by columnns, if empty then dimension id ascending order is used
}
```

Method:

```
POST /api/model/:model/run/:run/table/compare-id
```

For example:

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/run/Default/table/compare-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/compare-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:

Example 1. For `modelOne` output table `salarySex`:

- read `expr0` values from `[base]` run and from `[variant]` runs: "Default-4", "First Task Run_modelOne_other"
- calculate difference of `expr0` values between `[variant]` runs and `[base]` run

```
{
"Name":      "salarySex",
"Calculation": [
  {
    "Calculate": "expr0",
    "CalcId":    0,
    "Name":      "expr0",
    "IsAggr": false
  },
  {
    "Calculate": "expr0[variant] - expr0[base]",
    "CalcId":    12000,
    "Name":      "Diff_of_expr0",
    "IsAggr": false
  }
],
"Runs": [
  "Default-4",
  "First Task Run_modelOne_other"
]
}
```

Calculation must be done over output table expressions and can NOT include table accumulators. Output table expression can be from `[base]` or from `[variant]` model run, e.g.: `expr0[variant] - expr0[base]`. It is also possible to use to do a calcultion for each single (not between two runs), e.g.: `expr0`, or `100 * expr0 / expr1`. Calcultion must be a comparison formula between two runs `[base]` and `[variant]` or done on single run. You cannot mix comparison and sincle run calcultion, it is mutually exclusive, for exampel: thisi is an error: `expr1 + expr0[variant] - expr0[base]`. Calculation can include expression names, `+ - * /` operators and following functions:

- `SQRT` square root
- `ABS` absolute value
- `OM_IF` equivalent of `if .... then .... else ....`
- `OM_DIV_BY` wrap denominator

It is also possible to use parameter(s) in calculation, parameter must be a scalar of float or integer type. For example: `(Expr1[variant] - Expr1[base]) * param.Scale[base]`, where `param.Scale` is a value of scalar parameter `Scale` in `[base]` model run.

For more details please see: Model Output Expressions

Example 2:

- read `expr0` values from `[base]` run and from `[variant]` runs: "Default-4", "First Task Run_modelOne_other"
- calculate difference of `expr0` values between `[variant]` runs and `[base]`, adjusted by using parameter `StartingSeed` value from `base` run
- read only first 100 rows: `Offset: 0, Size: 100`
- apply WHERE filters and ORDER BY

```
{
"Name":      "salarySex",
"Calculation": [
  {
    "Calculate": "expr0",
    "CalcId":    0,
    "Name":      "expr0",
    "IsAggr": false
  },
  {
    "Calculate": "(expr0[variant] - expr0[base]) + param.StartingSeed[base] / 100",
    "CalcId":    12000,
    "Name":      "Diff_of_expr0_adjusted",
    "IsAggr": false
  }
],
"Runs": [
  "Default-4",
  "First Task Run_modelOne_other"
],
"Offset": 0,
"Size": 100,
"IsFullPage": true,
"IsSubId": true,
"SubId": 2,
"FilterById": [{
    "Name": "AgeGroup",
    "Op": "IN",
    "EnumIds": [100, 300]
  }, {
    "Name": "Province",
    "Op": "BETWEEN",
    "EnumIds": [1, 800]
  }
],
"OrderBy": [{
    "IndexOne": 2,
    "IsDesc": true
  }, {
    "IndexOne": 3,
    "IsDesc": 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, if size <= 0 then all rows selected
IsFullPage - (optional) if true then always return non-empty last page of data
IsSubId    - (optional) if true then select only single sub-value, default: all sub-values
SubId      - (optional) sub-value id to select if IsSubId is true
FilterById - (optional) conditions to filter dimension enum id's
OrderBy    - (optional) list of columns indexes (one based) to order by
```

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 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. First two columns are `run_id, calc_id`:

```
SELECT run_id, CalcId AS calc_id, dim0, dim1, ..., calc_value FROM .... ORDER BY 1, 2,...
```

Example:

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/run/Default/table/compare-id -d @test.json
```

JSON body (test.json):

```
{
  "Name":      "salarySex",
  "Calculation": [
    {
      "Calculate": "expr0",
      "CalcId":    0,
      "Name":      "expr0",
      "IsAggr": false
    },
    {
      "Calculate": "expr0[variant] - expr0[base]",
      "CalcId":    12000,
      "Name":      "Diff_of_expr0",
      "IsAggr": false
    }
  ],
  "Runs": [
    "Default-4",
    "First Task Run_modelOne_other"
  ]
}
```


Result:


```
> POST /api/model/modelOne/run/Default/table/compare-id HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/8.0.1
> Accept: */*
> Content-Type: application/json
> Content-Length: 363
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Date: Thu, 26 Oct 2023 02:55:09 GMT
< Transfer-Encoding: chunked
<
{
    "Page": [{
            "DimIds": [100, 0],
            "IsNull": false,
            "Value": 50,
            "CalcId": 0,
            "RunId": 201
        }, {
            "DimIds": [100, 1],
            "IsNull": false,
            "Value": 60,
            "CalcId": 0,
            "RunId": 201
        }, {
            "DimIds": [100, 800],
            "IsNull": false,
            "Value": 1,
            "CalcId": 0,
            "RunId": 201
        }, {
            "DimIds": [200, 0],
            "IsNull": false,
            "Value": 51.599999999999994,
            "CalcId": 0,
            "RunId": 201
        }, {
            "DimIds": [200, 1],
            "IsNull": false,
            "Value": 62,
            "CalcId": 0,
            "RunId": 201
        }, {
            "DimIds": [200, 800],
            "IsNull": false,
            "Value": 2,
            "CalcId": 0,
            "RunId": 201
        }, {
            "DimIds": [300, 0],
            "IsNull": false,
            "Value": 53.2,
            "CalcId": 0,
            "RunId": 201
        }, {
            "DimIds": [300, 1],
            "IsNull": false,
            "Value": 64,
            "CalcId": 0
```

"CalcId": 0,
"RunId": 201
}, {
"DimIds": [300, 800],
"IsNull": false,
"Value": 3,
"CalcId": 0,
"RunId": 201
}, {
"DimIds": [100, 0],
"IsNull": false,
"Value": 50,
"CalcId": 0,
"RunId": 202
}, {
"DimIds": [100, 1],
"IsNull": false,
"Value": 60,
"CalcId": 0,
"RunId": 202
}, {
"DimIds": [100, 800],
"IsNull": false,
"Value": 1201,
"CalcId": 0,
"RunId": 202
}, {
"DimIds": [200, 0],
"IsNull": false,
"Value": 51.599999999999994,
"CalcId": 0,
"RunId": 202
}, {
"DimIds": [200, 1],
"IsNull": false,
"Value": 62,
"CalcId": 0,
"RunId": 202
}, {
"DimIds": [200, 800],
"IsNull": false,
"Value": 1202,
"CalcId": 0,
"RunId": 202
}, {
"DimIds": [300, 0],
"IsNull": false,
"Value": 53.2,
"CalcId": 0,
"RunId": 202
}, {
"DimIds": [300, 1],
"IsNull": false,
"Value": 64,
"CalcId": 0,
"RunId": 202
}, {
"DimIds": [300, 800],
"IsNull": false,
"Value": 1203,
"CalcId": 0,
"RunId": 202
}, {
"DimIds": [100, 0],
"IsNull": false,
"Value": 0,
"CalcId": 12000,
"RunId": 202
}, {
"DimIds": [100, 1],
"IsNull": false,
"Value": 0,
"CalcId": 12000,
"RunId": 202
}, {
"DimIds": [100, 800],
"IsNull": false,
"Value": 1200,
"CalcId": 12000,
"RunId": 202
}, {
"DimIds": [200, 0],
"IsNull": false,
"Value": 0,
"CalcId": 12000,
"RunId": 202
}, {
"DimIds": [200, 1],

"IsNull": false,
"Value": 0,
"CalcId": 12000,
"RunId": 202
}, {
"DimIds": [200, 800],
"IsNull": false,
"Value": 1200,
"CalcId": 12000,
"RunId": 202
}, {
"DimIds": [300, 0],
"IsNull": false,
"Value": 0,
"CalcId": 12000,
"RunId": 202
}, {
"DimIds": [300, 1],
"IsNull": false,
"Value": 0,
"CalcId": 12000,
"RunId": 202
}, {
"DimIds": [300, 800],
"IsNull": false,
"Value": 1200,
"CalcId": 12000,
"RunId": 202
}, {
"DimIds": [100, 0],
"IsNull": false,
"Value": 225.6,
"CalcId": 0,
"RunId": 205
}, {
"DimIds": [100, 1],
"IsNull": false,
"Value": 272,
"CalcId": 0,
"RunId": 205
}, {
"DimIds": [100, 800],
"IsNull": false,
"Value": 1,
"CalcId": 0,
"RunId": 205
}, {
"DimIds": [200, 0],
"IsNull": false,
"Value": 232,
"CalcId": 0,
"RunId": 205
}, {
"DimIds": [200, 1],
"IsNull": false,
"Value": 280,
"CalcId": 0,
"RunId": 205
}, {
"DimIds": [200, 800],
"IsNull": false,
"Value": 2,
"CalcId": 0,
"RunId": 205
}, {
"DimIds": [300, 0],
"IsNull": false,
"Value": 238.39999999999998,
"CalcId": 0,
"RunId": 205
}, {
"DimIds": [300, 1],
"IsNull": false,
"Value": 288,
"CalcId": 0,
"RunId": 205
}, {
"DimIds": [300, 800],
"IsNull": false,
"Value": 3,
"CalcId": 0,
"RunId": 205
}, {
"DimIds": [100, 0],
"IsNull": false,
"Value": 175.6,
"CalcId": 12000,
"RunId": 205

```
        }, {
            "DimIds": [100, 1],
            "IsNull": false,
            "Value": 212,
            "CalcId": 12000,
            "RunId": 205
        }, {
            "DimIds": [100, 800],
            "IsNull": false,
            "Value": 0,
            "CalcId": 12000,
            "RunId": 205
        }, {
            "DimIds": [200, 0],
            "IsNull": false,
            "Value": 180.4,
            "CalcId": 12000,
            "RunId": 205
        }, {
            "DimIds": [200, 1],
            "IsNull": false,
            "Value": 218,
            "CalcId": 12000,
            "RunId": 205
        }, {
            "DimIds": [200, 800],
            "IsNull": false,
            "Value": 0,
            "CalcId": 12000,
            "RunId": 205
        }, {
            "DimIds": [300, 0],
            "IsNull": false,
            "Value": 185.2,
            "CalcId": 12000,
            "RunId": 205
        }, {
            "DimIds": [300, 1],
            "IsNull": false,
            "Value": 224,
            "CalcId": 12000,
            "RunId": 205
        }, {
            "DimIds": [300, 800],
            "IsNull": false,
            "Value": 0,
            "CalcId": 12000,
            "RunId": 205
        }
    ],
    "Layout": {
        "Offset": 0,
        "Size": 45,
        "IsLastPage": true,
        "IsFullPage": false
    }
}
```

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,
  "IsFullPage": true,
  "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, if size <= 0 then all rows selected
IsFullPage - (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
```

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:

```
{
    "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,
        "IsFullPage":false,
        "IsLastPage": false
    }
}
```


Example 2:

There are five filters applied to `Person` microdata rows:

- attribute `AgeGroup IN ["20-30", "40+"]`
- AND attribute `Sex = ["F"]`
- AND attribute `Pension > 44000`
- AND attribute `Pension < 77000`

JSON body:

```
{
  "Name": "Person",
  "Offset": 16,
  "Size": 8,
  "Filter": [{
      "Name": "AgeGroup",
      "Op": "IN",
      "Values": ["20-30", "40+"]
    }, {
      "Name": "Sex",
      "Op": "=",
      "Values": ["F"]
    }, {
      "Name": "IsOldAge",
      "Op": "!=",
      "Values": ["false"]
    }, {
      "Name": "Pension",
      "Op": ">",
      "Values": ["44000"]
    }, {
      "Name": "Pension",
      "Op": "<",
      "Values": ["77000"]
    }
  ],
  "OrderBy": [{
      "IndexOne": 5
    }, {
      "IndexOne": 6,
      "IsDesc": true
    }
  ]
}
```


Result:


```
{
  "Page": [{
      "Key": 2056,
      "Attr": [{
          "IsNull": false,
          "Value": 97
        }, {
          "IsNull": false,
          "Value": "40+"
        }, {
          "IsNull": false,
          "Value": "F"
        }, {
          "IsNull": false,
          "Value": 46290.3
        }, {
          "IsNull": false,
          "Value": 0
        }, {
          "IsNull": false,
          "Value": "L"
        }, {
          "IsNull": false,
          "Value": "Part"
        }, {
          "IsNull": false,
          "Value": true
        }, {
          "IsNull": false,
          "Value": 44086
        }
      ]
    }, {
      "Key": 281474976712711,
      "Attr": [{
          "IsNull": false,
          "Value": 97
        }, {
          "IsNull": false,
          "Value": "40+"
        }, {
          "IsNull": false,
          "Value": "F"
        }, {
          "IsNull": false,
          "Value": 46290.3
        }, {
          "IsNull": false,
```

```
        "Value": 0
      }, {
        "IsNull": false,
        "Value": "L"
      }, {
        "IsNull": false,
        "Value": "Part"
      }, {
        "IsNull": false,
        "Value": true
      }, {
        "IsNull": false,
        "Value": 44086
      }
    ]
  }, {
    "Key": 562949953423366,
    "Attr": [{
        "IsNull": false,
        "Value": 97
      }, {
        "IsNull": false,
        "Value": "40+"
      }, {
        "IsNull": false,
        "Value": "F"
      }, {
        "IsNull": false,
        "Value": 46290.3
      }, {
        "IsNull": false,
        "Value": 0
      }, {
        "IsNull": false,
        "Value": "L"
      }, {
        "IsNull": false,
        "Value": "Part"
      }, {
        "IsNull": false,
        "Value": true
      }, {
        "IsNull": false,
        "Value": 44086
      }
    ]
  }, {
    "Key": 844424930134021,
    "Attr": [{
        "IsNull": false,
        "Value": 97
      }, {
        "IsNull": false,
        "Value": "40+"
      }, {
        "IsNull": false,
        "Value": "F"
      }, {
        "IsNull": false,
        "Value": 46290.3
      }, {
        "IsNull": false,
        "Value": 0
      }, {
        "IsNull": false,
        "Value": "L"
      }, {
        "IsNull": false,
        "Value": "Part"
      }, {
        "IsNull": false,
        "Value": true
      }, {
        "IsNull": false,
        "Value": 44086
      }
    ]
  }, {
    "Key": 6146,
    "Attr": [{
        "IsNull": false,
        "Value": 87
      }, {
        "IsNull": false,
        "Value": "40+"
      }, {
        "IsNull": false,
        "Value": "F"
```

```
      "Value": "F"
    }, {
      "IsNull": false,
      "Value": 46296.6
    }, {
      "IsNull": false,
      "Value": 0
    }, {
      "IsNull": false,
      "Value": "L"
    }, {
      "IsNull": false,
      "Value": "Part"
    }, {
      "IsNull": false,
      "Value": true
    }, {
      "IsNull": false,
      "Value": 44092
    }
  ]
}, {
  "Key": 281474976716801,
  "Attr": [{
      "IsNull": false,
      "Value": 87
    }, {
      "IsNull": false,
      "Value": "40+"
    }, {
      "IsNull": false,
      "Value": "F"
    }, {
      "IsNull": false,
      "Value": 46296.6
    }, {
      "IsNull": false,
      "Value": 0
    }, {
      "IsNull": false,
      "Value": "L"
    }, {
      "IsNull": false,
      "Value": "Part"
    }, {
      "IsNull": false,
      "Value": true
    }, {
      "IsNull": false,
      "Value": 44092
    }
  ]
}, {
  "Key": 562949953427456,
  "Attr": [{
      "IsNull": false,
      "Value": 87
    }, {
      "IsNull": false,
      "Value": "40+"
    }, {
      "IsNull": false,
      "Value": "F"
    }, {
      "IsNull": false,
      "Value": 46296.6
    }, {
      "IsNull": false,
      "Value": 0
    }, {
      "IsNull": false,
      "Value": "L"
    }, {
      "IsNull": false,
      "Value": "Part"
    }, {
      "IsNull": false,
      "Value": true
    }, {
      "IsNull": false,
      "Value": 44092
    }
  ]
}, {
  "Key": 844424930138111,
  "Attr": [{
      "IsNull": false,
      "Value": 87
```

```
        }, {
          "IsNull": false,
          "Value": "40+"
        }, {
          "IsNull": false,
          "Value": "F"
        }, {
          "IsNull": false,
          "Value": 46296.6
        }, {
          "IsNull": false,
          "Value": 0
        }, {
          "IsNull": false,
          "Value": "L"
        }, {
          "IsNull": false,
          "Value": "Part"
        }, {
          "IsNull": false,
          "Value": true
        }, {
          "IsNull": false,
          "Value": 44092
        }
      ]
    }
  ],
  "Layout": {
    "Offset": 16,
    "Size": 8,
    "IsLastPage": false,
    "IsFullPage": 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,
  "IsFullPage": true,
  "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, if size <= 0 then all rows selected
IsFullPage - (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
```

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,
        "IsFullPage":false,
        "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,
        "IsFullPage":false,
        "IsLastPage": false
    }
}
```

Read aggregated microdata from model run

Read a "page" of aggregated microdata values from model run.

- Aggregate one or more microdata value attribute (float of integer type attribute). For example, two aggregations: `OM_AVG(Income)`, `OM_MAX(Salary + Pension)`.
- Group by one or more dimension attributes (enum-based or bool type attribute). For example, group by two dimension attributes: `AgeGroup`, `Sex`.
- 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 attribute(s) returned as enum codes. For boolean dimensions string value used, e.g.: "true".
- Method verb must be POST and Content-Type header "application/json".

Following aggregation functions avaliable:

- `OM_AVG` mean of accumulators sub-values
- `OM_SUM` sum of accumulators sub-values
- `OM_COUNT` count of accumulators sub-values (excluding NULL's)
- `OM_COUNT_IF` count values matching condition
- `OM_MAX` maximum of accumulators sub-values
- `OM_MIN` minimum of accumulators sub-values
- `OM_VAR` variance of accumulators sub-values
- `OM_SD` standard deviation of accumulators sub-values
- `OM_SE` standard error of accumulators sub-values
- `OM_CV` coefficient of variation of accumulators sub-values

It is also possible to use parameter(s) in calculation, parameter must be a scalar of float or integer type. For example: `OM_COUNT_IF(Income > param.High)`, where `param.High` is a value of scalar parameter `High` in that model run.

For more details please see: Model Output Expressions

JSON body POSTed to specify entity name, page size, row count, filters and row order. It is expected to be JSON representation of db.ReadCalculteMicroLayout structure from Go library. See also: db.ReadLayout structure from Go library.

```
// ReadCalculteMicroLayout describe microdata generation read layout, aggregation measures and group by attributes.
type ReadCalculteMicroLayout struct {
    ReadLayout           // entity name, run id, page size, where filters and order by
    CalculateMicroLayout // microdata aggregations
}

// CalculateMicroLayout describes aggregations of microdata.
//
// It can be comparison aggregations and/or calculation aggregations.
// Comparison aggregation must contain [base] and [variant] attribute(s), ex.: OM_AVG(Income[base] - Income[variant]).
// Calculation aggregation is attribute(s) aggregation expression, ex.: OM_MAX(Income) / OM_MIN(Salary).
type CalculateMicroLayout struct {
    Calculation []CalculateLayout // aggregation measures, ex.: OM_MIN(Salary), OM_AVG(Income[base] - Income[variant])
    GroupBy     []string          // attributes to group by
}

// CalculateLayout describes calculation expression for parameters, output table values or microdata entity.
// It can be comparison calculation for multiple model runs, ex.: Expr0[base] - Expr0[variant].
type CalculateLayout struct {
    Calculate string // expression to calculate, ex.: Expr0[base] - Expr0[variant]
    CalcId    int    // calculated expression id, calc_id column in csv,     ex.: 0, 12000, 24000
    Name      string // calculated expression name, calc_name column in csv, ex.: Expr0, AVG_Expr0, RATIO_Expro0
}

// ReadLayout describes source and size of data page to read input parameter, output table values or microdata.
//
// Row filters combined by AND and allow to select dimension or attribute items,
// it can be enum codes or enum id's, ex.: dim0 = 'CA' AND dim1 IN (2010, 2011, 2012)
type ReadLayout struct {
    Name           string           // parameter name, output table name or entity microdata name
    FromId         int              // run id or set id to select input parameter, output table values or microdata from
    ReadPageLayout                  // read page first row offset, size and last page flag
    Filter         []FilterColumn   // dimension or attribute filters, final WHERE does join all filters by AND
    FilterById     []FilterIdColumn // dimension or attribute filters by enum ids, final WHERE does join filters by AND
    OrderBy        []OrderByColumn  // order by columnns, if empty then dimension id ascending order is used
}
```

Methods:

```
POST /api/model/:model/run/:run/microdata/calc
```

For example:

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/calc -d @read_m1_person_calc_1.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:

Example 1: Aggregate `Person` entity to get `OM_AVG()` average `Income` value and group it by `AgeGroup , Sex` dimension attributes.

```
{
   "Name": "Person",
   "Calculation": [{
         "Calculate": "OM_AVG(Income)",
         "CalcId": 2401,
         "Name": "Avg_Income"
      }
   ],
   "GroupBy": [
      "AgeGroup",
      "Sex"
   ]
}
```

Example 2.

- aggregate `Person` entity
- calculate two values:
  - `OM_AVG()` average of `Income` value, adjusted by using parameter `StartingSeed` values
  - `OM_AVG()` average of `Salary + Pension` value, adjusted by using parameter `StartingSeed` values
- and group it by `AgeGroup , Sex` dimension attributes
- filter only rows where:
  - dimension `AgeGroup IN ["20-30", "40+"]`
  - and dimension `Sex = "F"`
  - and value of `Avg_Income_adjusted > 8500000`
  - and value of `Avg_Salary_Pension_adjusted < 85000`

```
{
"Name": "Person",
"Calculation": [{
      "Calculate": "OM_AVG(Income) * (param.StartingSeed / 100)",
      "CalcId": 2401,
      "Name": "Avg_Income_adjusted"
   }, {
      "Calculate": "OM_AVG(Salary + Pension + param.StartingSeed)",
      "CalcId": 2404,
      "Name": "Avg_Salary_Pension_adjusted"
   }
],
"GroupBy": [
   "AgeGroup",
   "Sex"
],
"Offset": 0,
"Size": 100,
"IsFullPage": true,
"Filter": [{
      "Name": "AgeGroup",
      "Op": "IN",
      "Values": ["20-30", "40+"]
   }, {
      "Name": "Sex",
      "Op": "=",
      "Values": ["F"]
   }, {
      "Name": "Avg_Income_adjusted",
      "Op": ">",
      "Values": ["8500000"]
   }, {
      "Name": "Avg_Salary_Pension_adjusted",
      "Op": "<",
      "Values": ["85000"]
   }
],
"OrderBy": [{
      "IndexOne": 2,
      "IsDesc": true
   }, {
      "IndexOne": 3,
      "IsDesc": true
   }
]
}
```

```
Name       - (required) entity name
Offset     - (optional) zero-based start row to select aggreagted microdata values
Size       - (optional) max row count to select rows, if size <= 0 then all rows selected
IsFullPage - (optional) if true then always return non-empty last page of data
Filter     - (optional) conditions to filter dimension attributes
OrderBy    - (optional) list of columns indexes (one based) to order by
```

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. First two columns are `run_id, calc_id`:

```
SELECT run_id, CalcId AS calc_id, AgeGroup, Sex, ..., calc_value FROM .... 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...]
}
```

Result:


```
{
  "Page": [{
      "Attr": [{
          "IsNull": false,
          "Value": "40+"
        }, {
          "IsNull": false,
          "Value": "F"
        }, {
          "IsNull": false,
          "Value": 69934.18
        }
      ],
      "CalcName": "Avg_Salary_Pension_adjusted",
      "RunDigest": "703d8b78039d69b795ab2e601c32b789"
    }, {
      "Attr": [{
          "IsNull": false,
          "Value": "20-30"
        }, {
          "IsNull": false,
          "Value": "F"
        }, {
          "IsNull": false,
          "Value": 8897709.41
        }
      ],
      "CalcName": "Avg_Income_adjusted",
      "RunDigest": "703d8b78039d69b795ab2e601c32b789"
    }
  ],
  "Layout": {
    "Offset": 0,
    "Size": 2,
    "IsLastPage": true,
    "IsFullPage": true
  }
}
```

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

Read a "page" of aggregated microdata values from model run.

- Aggregate one or more microdata value attribute (float of integer type attribute). For example, two aggregations: `OM_AVG(Income)`, `OM_MAX(Salary + Pension)`.
- Group by one or more dimension attributes (enum-based or bool type attribute). For example, group by two dimension attributes: `AgeGroup`, `Sex`.
- 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.
- Method verb must be POST and Content-Type header "application/json".

Following aggregation functions avaliable:

- `OM_AVG` mean of accumulators sub-values
- `OM_SUM` sum of accumulators sub-values
- `OM_COUNT` count of accumulators sub-values (excluding NULL's)
- `OM_COUNT_IF` count values matching condition
- `OM_MAX` maximum of accumulators sub-values
- `OM_MIN` minimum of accumulators sub-values
- `OM_VAR` variance of accumulators sub-values
- `OM_SD` standard deviation of accumulators sub-values
- `OM_SE` standard error of accumulators sub-values
- `OM_CV` coefficient of variation of accumulators sub-values

It is also possible to use parameter(s) in calculation, parameter must be a scalar of float or integer type. For example: `OM_COUNT_IF(Income > param.High)`, where `param.High` is a value of scalar parameter `High` in that model run.

For more details please see: Model Output Expressions

JSON body POSTed to specify entity name, page size, row count, filters and row order. It is expected to be JSON representation of db.ReadCalculteMicroLayout structure from Go library. See also: db.ReadLayout structure from Go library.

```
// ReadCalculteMicroLayout describe microdata generation read layout, aggregation measures and group by attributes.
type ReadCalculteMicroLayout struct {
    ReadLayout          // entity name, run id, page size, where filters and order by
    CalculateMicroLayout // microdata aggregations
}

// CalculateMicroLayout describes aggregations of microdata.
//
// It can be comparison aggregations and/or calculation aggregations.
// Comparison aggregation must contain [base] and [variant] attribute(s), ex.: OM_AVG(Income[base] - Income[variant]).
// Calculation aggregation is attribute(s) aggregation expression, ex.: OM_MAX(Income) / OM_MIN(Salary).
type CalculateMicroLayout struct {
    Calculation []CalculateLayout // aggregation measures, ex.: OM_MIN(Salary), OM_AVG(Income[base] - Income[variant])
    GroupBy     []string          // attributes to group by
}

// CalculateLayout describes calculation expression for parameters, output table values or microdata entity.
// It can be comparison calculation for multiple model runs, ex.: Expr0[base] - Expr0[variant].
type CalculateLayout struct {
    Calculate string // expression to calculate, ex.: Expr0[base] - Expr0[variant]
    CalcId    int    // calculated expression id, calc_id column in csv,     ex.: 0, 12000, 24000
    Name      string // calculated expression name, calc_name column in csv, ex.: Expr0, AVG_Expr0, RATIO_Expro0
}

// ReadLayout describes source and size of data page to read input parameter, output table values or microdata.
//
// Row filters combined by AND and allow to select dimension or attribute items,
// it can be enum codes or enum id's, ex.: dim0 = 'CA' AND dim1 IN (2010, 2011, 2012)
type ReadLayout struct {
    Name           string           // parameter name, output table name or entity microdata name
    FromId         int              // run id or set id to select input parameter, output table values or microdata from
    ReadPageLayout                  // read page first row offset, size and last page flag
    Filter         []FilterColumn   // dimension or attribute filters, final WHERE does join all filters by AND
    FilterById     []FilterIdColumn // dimension or attribute filters by enum ids, final WHERE does join filters by AND
    OrderBy        []OrderByColumn  // order by columnns, if empty then dimension id ascending order is used
}
```

Methods:

```
POST /api/model/:model/run/:run/microdata/calc-id
```

For example:

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/calc-id -d @read_m1_person_calc_1.jso
n
```

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:

Example 1: Aggregate `Person` entity to get `OM_AVG()` average `Income` value and group it by `AgeGroup , Sex` dimension attributes.

```
{
   "Name": "Person",
   "Calculation": [{
         "Calculate": "OM_AVG(Income)",
         "CalcId": 2401,
         "Name": "Avg_Income"
      }
   ],
   "GroupBy": [
      "AgeGroup",
      "Sex"
   ]
}
```

Example 2.

- aggregate `Person` entity
- calculate two values:
  - `OM_AVG()` average of `Income` value, adjusted by using parameter `StartingSeed` values
  - `OM_AVG()` average of `Salary + Pension` value, adjusted by using parameter `StartingSeed` values
- and group it by `AgeGroup , Sex` dimension attributes
- filter only rows where:
  - enum id's of dimension `AgeGroup IN [20, 40]`
  - and enum id's of dimension `Sex = 1`

```
{
"Name": "Person",
"Calculation": [{
      "Calculate": "OM_AVG(Income) * (param.StartingSeed / 100)",
      "CalcId": 2401,
      "Name": "Avg_Income_adjusted"
   }, {
      "Calculate": "OM_AVG(Salary + Pension + param.StartingSeed)",
      "CalcId": 2404,
      "Name": "Avg_Salary_Pension_adjusted"
   }
],
"GroupBy": [
   "AgeGroup",
   "Sex"
],
"Offset": 0,
"Size": 100,
"IsFullPage": 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) entity name
Offset     - (optional) zero-based start row to select aggreagted microdata values
Size       - (optional) max row count to select rows, if size <= 0 then all rows selected
IsFullPage - (optional) if true then always return non-empty last page of data
FilterById - (optional) conditions to filter enum id's of dimension attributes
OrderBy    - (optional) list of columns indexes (one based) to order by
```

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. First two columns are `run_id, calc_id`:

```
SELECT run_id, CalcId AS calc_id, AgeGroup, Sex, ..., calc_value FROM .... 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...]
}
```

Result:

```
*   Trying [::1]:4040...
* Connected to localhost (::1) port 4040
> POST /api/model/modelOne/run/Microdata%20in%20database/microdata/calc-id HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/8.4.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 753
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Date: Fri, 29 Dec 2023 03:53:37 GMT
< Content-Length: 609
<
{
    "Page": [{
            "Attr": [{
                    "IsNull": false,
                    "Value": 40
                }, {
                    "IsNull": false,
                    "Value": 1
                }, {
                    "IsNull": false,
                    "Value": 57665830.54215979
                }
            ],
            "CalcId": 2404,
            "RunId": 219
        }, {
            "Attr": [{
                    "IsNull": false,
                    "Value": 20
                }, {
                    "IsNull": false,
                    "Value": 1
                }, {
                    "IsNull": false,
                    "Value": 100657151.25
                }
            ],
            "CalcId": 2404,
            "RunId": 219
        }, {
            "Attr": [{
                    "IsNull": false,
                    "Value": 40
                }, {
                    "IsNull": false,
                    "Value": 1
                }, {
                    "IsNull": false,
                    "Value": 71069306.57187325
                }
            ],
            "CalcId": 2401,
            "RunId": 219
        }, {
            "Attr": [{
                    "IsNull": false,
                    "Value": 20
                }, {
                    "IsNull": false,
                    "Value": 1
                }, {
                    "IsNull": false,
                    "Value": 134209535
                }
            ],
            "CalcId": 2401,
            "RunId": 219
        }
    ],
    "Layout": {
        "Offset": 0,
        "Size": 4,
        "IsLastPage": true,
        "IsFullPage": true
    }
}
```

Read microdata run comparison

Read a "page" of microdata values and compare model runs.

Compare `[base]` and `[variant]` model runs microdata value attributes (float of integer type attributes), group it by dimension attributes (enum-based or bool type attributes).

- Compare one or more microdata value attributes (float of integer type attribute). For example, two comparisons: `OM_AVG(Income[variant] - Income[base]) , OM_MAX( 100 * (Salary[variant] + Pension[variant]) / Income[base])`.
- It is also posiible to include aggreagted value attribute(s) for each single run, for example: `OM_MAX(Salary) , OM_MIN(Pension)`.
- Group by one or more dimension attributes (enum-based or bool type attribute). For example, group by two dimension attributes: `AgeGroup , Sex`.
- 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 attribute(s) returned as enum codes. For boolean dimensions string value used, e.g.: "true".
- Method verb must be POST and Content-Type header "application/json".

Following aggregation functions avaliable:

- `OM_AVG` mean of accumulators sub-values
- `OM_SUM` sum of accumulators sub-values
- `OM_COUNT` count of accumulators sub-values (excluding NULL's)
- `OM_COUNT_IF` count values matching condition
- `OM_MAX` maximum of accumulators sub-values
- `OM_MIN` minimum of accumulators sub-values
- `OM_VAR` variance of accumulators sub-values
- `OM_SD` standard deviation of accumulators sub-values
- `OM_SE` standard error of accumulators sub-values
- `OM_CV` coefficient of variation of accumulators sub-values

It is also possible to use parameter(s) in calculation, parameter must be a scalar of float or integer type. For example: `OM_COUNT_IF((Income[variant - Income[base]) > param.High[base])`, where `param.High[base]` is a value of scalar parameter `High` in `[base]` model run.

For more details please see: Model Output Expressions

JSON body POSTed to specify entity name, page size, row count, filters and row order. It is expected to be JSON representation of db.ReadCompareMicroLayout structure from Go library. See also: db.ReadLayout structure from Go library.

```
// ReadCompareMicroLayout to compare microdata runs with base run using multiple comparison aggregations and/or calculation aggregations.
//
// Comparison aggregation must contain [base] and [variant] attribute(s), ex.: OM_AVG(Income[base] - Income[variant]).
// Calculation aggregation is attribute(s) aggregation expression, ex.: OM_MAX(Income) / OM_MIN(Salary).
type ReadCompareMicroLayout struct {
    ReadCalculteMicroLayout          // aggregation measures and group by attributes
    Runs                    []string // runs to compare: list of digest, stamp or name
}

// ReadCalculteMicroLayout describe microdata generation read layout, aggregation measures and group by attributes.
type ReadCalculteMicroLayout struct {
    ReadLayout           // entity name, run id, page size, where filters and order by
    CalculateMicroLayout // microdata aggregations
}

// CalculateMicroLayout describes aggregations of microdata.
//
// It can be comparison aggregations and/or calculation aggregations.
// Comparison aggregation must contain [base] and [variant] attribute(s), ex.: OM_AVG(Income[base] - Income[variant]).
// Calculation aggregation is attribute(s) aggregation expression, ex.: OM_MAX(Income) / OM_MIN(Salary).
type CalculateMicroLayout struct {
    Calculation []CalculateLayout // aggregation measures, ex.: OM_MIN(Salary), OM_AVG(Income[base] - Income[variant])
    GroupBy     []string          // attributes to group by
}

// CalculateLayout describes calculation expression for parameters, output table values or microdata entity.
// It can be comparison calculation for multiple model runs, ex.: Expr0[base] - Expr0[variant].
type CalculateLayout struct {
    Calculate string // expression to calculate, ex.: Expr0[base] - Expr0[variant]
    CalcId    int    // calculated expression id, calc_id column in csv,     ex.: 0, 12000, 24000
    Name      string // calculated expression name, calc_name column in csv, ex.: Expr0, AVG_Expr0, RATIO_Expro0
}

// ReadLayout describes source and size of data page to read input parameter, output table values or microdata.
//
// Row filters combined by AND and allow to select dimension or attribute items,
// it can be enum codes or enum id's, ex.: dim0 = 'CA' AND dim1 IN (2010, 2011, 2012)
type ReadLayout struct {
    Name           string           // parameter name, output table name or entity microdata name
    FromId         int              // run id or set id to select input parameter, output table values or microdata from
    ReadPageLayout                  // read page first row offset, size and last page flag
    Filter         []FilterColumn   // dimension or attribute filters, final WHERE does join all filters by AND
    FilterById     []FilterIdColumn // dimension or attribute filters by enum ids, final WHERE does join filters by AND
    OrderBy        []OrderByColumn  // order by columnns, if empty then dimension id ascending order is used
}
```

Methods:

```
POST /api/model/:model/run/:run/microdata/compare
```

For example:

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/compare -d @read_m1_person_cmp_1.j
son
```

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:

Example 1: Compare `Person` entity between `[base]` model run and `[variant]` model run: `Microdata other in database` by `OM_AVG()` average `Income[variant] - Income[base]` value and group it by `AgeGroup , Sex` dimension attributes.

```
{
   "Name": "Person",
   "Calculation": [{
         "Calculate": "OM_AVG(Income[variant] - Income[base])",
         "CalcId": 2401,
         "Name": "Avg_Income"
      }
   ],
   "GroupBy": [
      "AgeGroup",
      "Sex"
   ],
   "Runs": [
      "Microdata other in database"
   ]
}
```

Example 2.

- compare `Person` entity
- between `[base]` model run and `[variant]` model run: `Microdata other in database`
- calculate two values:
  - `OM_AVG()` average of `Income[variant] - Income[base]` value, adjusted by using parameter `StartingSeed` values
  - `OM_AVG()` average of `Salary + Pension` value, adjusted by using parameter `StartingSeed` values
- and group it by `AgeGroup , Sex` dimension attributes
- filter only rows where:
  - dimension `AgeGroup IN ["20-30", "40+"]`
  - and dimension `Sex = "F"`
  - and value of `Avg_Income_adjusted > 65000`
  - and value of `Avg_Salary_Pension_adjusted < 75000`

```
{
"Name": "Person",
"Calculation": [{
      "Calculate": "OM_AVG((Income[variant] - Income[base]) * (param.StartingSeed[variant] - param.StartingSeed[base]))",
      "CalcId": 2401,
      "Name": "Avg_Income_adjusted"
   }, {
      "Calculate": "param.StartingSeed + OM_AVG(Salary + Pension)",
      "CalcId": 2408,
      "Name": "Avg_Salary_Pension_adjusted"
   }
],
"GroupBy": [
   "AgeGroup",
   "Sex"
],
"Runs": [
   "Microdata other in database"
],
"Offset": 0,
"Size": 100,
"IsFullPage": true,
"Filter": [{
      "Name": "AgeGroup",
      "Op": "IN",
      "Values": ["20-30", "40+"]
   }, {
      "Name": "Sex",
      "Op": "=",
      "Values": ["F"]
   }, {
      "Name": "Avg_Income_adjusted",
      "Op": ">",
      "Values": ["65000"]
   }, {
      "Name": "Avg_Salary_Pension_adjusted",
      "Op": "<",
      "Values": ["75000"]
   }
]
}
```

```
Name       - (required) entity name
Offset     - (optional) zero-based start row to select aggreagted microdata values
Size       - (optional) max row count to select rows, if size <= 0 then all rows selected
IsFullPage - (optional) if true then always return non-empty last page of data
Filter     - (optional) conditions to filter dimension attributes
OrderBy    - (optional) list of columns indexes (one based) to order by
```

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. First two columns are `run_id, calc_id`:

```
SELECT run_id, CalcId AS calc_id, AgeGroup, Sex, ..., calc_value FROM .... 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...]
}
```

Result:


```
{
  "Page": [{
      "Attr": [{
          "IsNull": false,
          "Value": "40+"
        }, {
          "IsNull": false,
          "Value": "F"
        }, {
          "IsNull": false,
          "Value": 69934.18659670698
        }
      ],
      "CalcName": "Avg_Salary_Pension_adjusted",
      "RunDigest": "703d8b78039d69b795ab2e601c32b789"
    }, {
      "Attr": [{
          "IsNull": false,
          "Value": "20-30"
        }, {
          "IsNull": false,
          "Value": "F"
        }, {
          "IsNull": false,
          "Value": 920051.7871559632
        }
      ],
      "CalcName": "Avg_Income_adjusted",
      "RunDigest": "c8382ce22a004bf86d83000aa022d45b"
    }, {
      "Attr": [{
          "IsNull": false,
          "Value": "40+"
        }, {
          "IsNull": false,
          "Value": "F"
        }, {
          "IsNull": false,
          "Value": 65848.00593810345
        }
      ],
      "CalcName": "Avg_Salary_Pension_adjusted",
      "RunDigest": "c8382ce22a004bf86d83000aa022d45b"
    }
  ],
  "Layout": {
    "Offset": 0,
    "Size": 3,
    "IsLastPage": true,
    "IsFullPage": true
  }
}
```

Read microdata run comparison (enum id's)

Read a "page" of microdata values and compare model runs.

Compare `[base]` and `[variant]` model runs microdata value attributes (float of integer type attributes), group it by dimension attributes (enum-based or bool type attributes).

- Compare one or more microdata value attributes (float of integer type attribute). For example, two comparisons: `OM_AVG(Income[variant] - Income[base]) , OM_MAX( 100 * (Salary[variant] + Pension[variant]) / Income[base])` .
- It is also posiible to include aggreagted value attribute(s) for each single run, for example: `OM_MAX(Salary) , OM_MIN(Pension)` .
- Group by one or more dimension attributes (enum-based or bool type attribute). For example, group by two dimension attributes: `AgeGroup , Sex` .
- 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.
- Method verb must be POST and Content-Type header "application/json".

Following aggregation functions avaliable:

- `OM_AVG` mean of accumulators sub-values
- `OM_SUM` sum of accumulators sub-values
- `OM_COUNT` count of accumulators sub-values (excluding NULL's)
- `OM_COUNT_IF` count values matching condition
- `OM_MAX` maximum of accumulators sub-values
- `OM_MIN` minimum of accumulators sub-values
- `OM_VAR` variance of accumulators sub-values
- `OM_SD` standard deviation of accumulators sub-values
- `OM_SE` standard error of accumulators sub-values
- `OM_CV` coefficient of variation of accumulators sub-values

It is also possible to use parameter(s) in calculation, parameter must be a scalar of float or integer type. For example: `OM_COUNT_IF((Income[variant - Income[base]) > param.High[base])` , where `param.High[base]` is a value of scalar parameter `High` in `[base]` model run.

For more details please see: Model Output Expressions

JSON body POSTed to specify entity name, page size, row count, filters and row order. It is expected to be JSON representation of db.ReadCompareMicroLayout structure from Go library. See also: db.ReadLayout structure from Go library.

```
// ReadCompareMicroLayout to compare microdata runs with base run using multiple comparison aggregations and/or calculation aggregations.
//
// Comparison aggregation must contain [base] and [variant] attribute(s), ex.: OM_AVG(Income[base] - Income[variant]).
// Calculation aggregation is attribute(s) aggregation expression, ex.: OM_MAX(Income) / OM_MIN(Salary).
type ReadCompareMicroLayout struct {
    ReadCalculteMicroLayout          // aggregation measures and group by attributes
    Runs                    []string // runs to compare: list of digest, stamp or name
}

// ReadCalculteMicroLayout describe microdata generation read layout, aggregation measures and group by attributes.
type ReadCalculteMicroLayout struct {
    ReadLayout           // entity name, run id, page size, where filters and order by
    CalculateMicroLayout // microdata aggregations
}

// CalculateMicroLayout describes aggregations of microdata.
//
// It can be comparison aggregations and/or calculation aggregations.
// Comparison aggregation must contain [base] and [variant] attribute(s), ex.: OM_AVG(Income[base] - Income[variant]).
// Calculation aggregation is attribute(s) aggregation expression, ex.: OM_MAX(Income) / OM_MIN(Salary).
type CalculateMicroLayout struct {
    Calculation []CalculateLayout // aggregation measures, ex.: OM_MIN(Salary), OM_AVG(Income[base] - Income[variant])
    GroupBy     []string          // attributes to group by
}

// CalculateLayout describes calculation expression for parameters, output table values or microdata entity.
// It can be comparison calculation for multiple model runs, ex.: Expr0[base] - Expr0[variant].
type CalculateLayout struct {
    Calculate string // expression to calculate, ex.: Expr0[base] - Expr0[variant]
    CalcId    int    // calculated expression id, calc_id column in csv,     ex.: 0, 12000, 24000
    Name      string // calculated expression name, calc_name column in csv, ex.: Expr0, AVG_Expr0, RATIO_Expro0
}

// ReadLayout describes source and size of data page to read input parameter, output table values or microdata.
//
// Row filters combined by AND and allow to select dimension or attribute items,
// it can be enum codes or enum id's, ex.: dim0 = 'CA' AND dim1 IN (2010, 2011, 2012)
type ReadLayout struct {
    Name           string           // parameter name, output table name or entity microdata name
    FromId         int              // run id or set id to select input parameter, output table values or microdata from
    ReadPageLayout                  // read page first row offset, size and last page flag
    Filter         []FilterColumn   // dimension or attribute filters, final WHERE does join all filters by AND
    FilterById     []FilterIdColumn // dimension or attribute filters by enum ids, final WHERE does join filters by AND
    OrderBy        []OrderByColumn  // order by columnns, if empty then dimension id ascending order is used
}
```

Methods:

```
POST /api/model/:model/run/:run/microdata/compare-id
```

For example:

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/compare-id -d @read_m1_person_cmp_1.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:

Example 1: Compare `Person` entity between `[base]` model run and `[variant]` model run: `Microdata other in database` by `OM_AVG()` average `Income[variant] - Income[base]` value and group it by `AgeGroup , Sex` dimension attributes.

```
{
   "Name": "Person",
   "Calculation": [{
         "Calculate": "OM_AVG(Income[variant] - Income[base])",
         "CalcId": 2401,
         "Name": "Avg_Income"
      }
   ],
   "GroupBy": [
      "AgeGroup",
      "Sex"
   ],
   "Runs": [
      "Microdata other in database"
   ]
}
```

Example 2.

- compare `Person` entity
- between `[base]` model run and `[variant]` model run: `Microdata other in database`
- calculate two values:
  - `OM_AVG()` average of `Income[variant] - Income[base]` value, adjusted by using parameter `StartingSeed` values
  - `OM_AVG()` average of `Salary + Pension` value, adjusted by using parameter `StartingSeed` values
- and group it by `AgeGroup , Sex` dimension attributes
- filter only rows where:
  - enum id's of dimension `AgeGroup IN [20, 40]`
  - and enum id's of dimension `Sex = 1`

```
{
"Name": "Person",
"Calculation": [{
      "Calculate": "OM_AVG((Income[variant] - Income[base]) * (param.StartingSeed[variant] - param.StartingSeed[base]))",
      "CalcId": 2401,
      "Name": "Avg_Income_adjusted"
   }, {
      "Calculate": "param.StartingSeed + OM_AVG(Salary + Pension)",
      "CalcId": 2408,
      "Name": "Avg_Salary_Pension_adjusted"
   }
],
"GroupBy": [
   "AgeGroup",
   "Sex"
],
"Runs": [
   "Microdata other in database"
],
"Offset": 0,
"Size": 100,
"IsFullPage": true,
"FilterById": [{
      "Name": "AgeGroup",
      "Op": "IN",
      "EnumIds": [20, 40]
   }, {
      "Name": "Sex",
      "Op": "=",
      "EnumIds": [1]
   }
]
}
```

```
Name       - (required) entity name
Offset     - (optional) zero-based start row to select aggreagted microdata values
Size       - (optional) max row count to select rows, if size <= 0 then all rows selected
IsFullPage - (optional) if true then always return non-empty last page of data
FilterById - (optional) conditions to filter enum id's of dimension attributes
OrderBy    - (optional) list of columns indexes (one based) to order by
```

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
```

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. First two columns are `run_id, calc_id`:

```
SELECT run_id, CalcId AS calc_id, AgeGroup, Sex, ..., calc_value FROM .... 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...]
}
```

Result:

```
*   Trying [::1]:4040...
* Connected to localhost (::1) port 4040
> POST /api/model/modelOne/run/Microdata%20in%20database/microdata/compare-id HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/8.4.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 685
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Date: Fri, 29 Dec 2023 04:28:01 GMT
< Content-Length: 876
<
{
    "Page": [{
            "Attr": [{
                    "IsNull": false,
                    "Value": 20
                }, {
                    "IsNull": false,
                    "Value": 1
                }, {
                    "IsNull": false,
                    "Value": 100657151.25
                }
            ],
            "CalcId": 2408,
            "RunId": 219
        }, {
            "Attr": [{
                    "IsNull": false,
                    "Value": 40
                }, {
                    "IsNull": false,
                    "Value": 1
                }, {
                    "IsNull": false,
                    "Value": 57665830.54215979
                }
            ],
            "CalcId": 2408,
            "RunId": 219
        }, {
            "Attr": [{
                    "IsNull": false,
                    "Value": 20
                }, {
                    "IsNull": false,
                    "Value": 1
                }, {
                    "IsNull": false,
```

```
                "Value": -67112960
            }
        ],
        "CalcId": 2401,
        "RunId": 221
    }, {
        "Attr": [{
                "IsNull": false,
                "Value": 40
            }, {
                "IsNull": false,
                "Value": 1
            }, {
                "IsNull": false,
                "Value": -35538991.09092548
            }
        ],
        "CalcId": 2401,
        "RunId": 221
    }, {
        "Attr": [{
                "IsNull": false,
                "Value": 20
            }, {
                "IsNull": false,
                "Value": 1
            }, {
                "IsNull": false,
                "Value": 50322431.25
            }
        ],
        "CalcId": 2408,
        "RunId": 221
    }, {
        "Attr": [{
                "IsNull": false,
                "Value": 40
            }, {
                "IsNull": false,
                "Value": 1
            }, {
                "IsNull": false,
                "Value": 28829395.64922502
            }
        ],
        "CalcId": 2408,
        "RunId": 221
    }
],
"Layout": {
    "Offset": 0,
    "Size": 6,
    "IsLastPage": true,
    "IsFullPage": true
}
}
```

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, if count <= 0 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, if count <= 0 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, if count <= 0 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 calculated expression(s) from model run

Read a "page" of output table calculated expression(s) values from model run.

Read output table expression(s) and calculate additrional measures. Measures calculated as one of the following:

- for each table expression calculate one of: avg, sum, count, max, min, var, sd, se, cv
- as arbitrary aggreagated expressions provided as comma separated list

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/calc/:calc
GET /api/model/:model/run/:run/table/:name/calc/:calc/start/:start
GET /api/model/:model/run/:run/table/:name/calc/:calc/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
```

```
:calc - (required) name of additional measure to calculate
```

Additional measure must be one of:

- `avg` mean of expression sub-values
- `sum` sum of expression sub-values
- `count` count of expression sub-values (excluding NULL's)
- `max` maximum of expression sub-values
- `min` minimum of expression sub-values
- `var` variance of expression sub-values
- `sd` standard deviation of expression sub-values
- `se` standard error of expression sub-values
- `cv` coefficient of variation of expression sub-values

Or a list of comma-separated aggreagated expressions, for example: OM_AVG(acc0) , 2 * SQRT(OM_SUM(acc1) - OM_SD(acc0))

It is also possible to use parameter(s) in calculation, parameter must be a scalar of float or integer type. For example: `OM_COUNT_IF(acc1 > param.High)`, where `param.High` is a value of scalar parameter `High` in that model run.

Following aggregation functions avaliable:

- `OM_AVG` mean of accumulators sub-values
- `OM_SUM` sum of accumulators sub-values
- `OM_COUNT` count of accumulators sub-values (excluding NULL's)
- `OM_COUNT_IF` count values matching condition
- `OM_MAX` maximum of accumulators sub-values
- `OM_MIN` minimum of accumulators sub-values
- `OM_VAR` variance of accumulators sub-values
- `OM_SD` standard deviation of accumulators sub-values
- `OM_SE` standard error of accumulators sub-values
- `OM_CV` coefficient of variation of accumulators sub-values

For more details please see: Model Output Expressions

```
:start - (optional) start "page" row number, zero-based.
:count - (optional) "page" size, number of rows to select, if count <= 0 then all rows selected.
```

Call examples:

```
http://localhost:4040/api/model/RiskPaths/run/RiskPaths_Default/table/T04_FertilityRatesByAgeGroup/calc/se
http://localhost:4040/api/model/_201208171604590148_/run/f172e98da17beb058f30f11768053456/table/salarySex/calc/se/start/2
http://localhost:4040/api/model/_201208171604590148_/run/f172e98da17beb058f30f11768053456/table/salarySex/calc/se/start/2/count/4

http://localhost:4040/api/model/_201208171604590148_/run/2019_01_17_19_59_52_998/table/salarySex/calc/se

http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/calc/avg
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/calc/sum
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/calc/count
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/calc/min
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/calc/max
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/calc/var
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/calc/sd
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/calc/se
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/calc/cv

http://localhost:4040/api/model/modelOne/run/Default-4/table/salarySex/calc/OM_AVG(acc0),2*SQRT(OM_SUM(acc1)-OM_SD(acc0))
http://localhost:4040/api/model/modelOne/run/Default-4/table/salarySex/calc/OM_COUNT_IF(acc0%3Cparam.StartingSeed)
```

Note: `OM_COUNT_IF(acc0%3Cparam.StartingSeed)` is URL encoded: `OM_COUNT_IF(acc0<param.StartingSeed)`

Example:

```
http://localhost:4040/api/model/RiskPaths/run/RiskPaths_Default_4/table/T04_FertilityRatesByAgeGroup/calc/se
```


```
[
  {
    "Dims": [
      "(-∞,15)",
      "US_NEVER_IN_UNION"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "(-∞,15)",
      "US_FIRST_UNION_PERIOD1"
    ],
    "IsNull": true,
    "Value": 0,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "(-∞,15)"
```

```
    "(-∞,15)",
    "US_FIRST_UNION_PERIOD2"
  ],
  "IsNull": true,
  "Value": 0,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "(-∞,15)",
    "US_AFTER_FIRST_UNION"
  ],
  "IsNull": true,
  "Value": 0,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "(-∞,15)",
    "US_SECOND_UNION"
  ],
  "IsNull": true,
  "Value": 0,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "(-∞,15)",
    "US_AFTER_SECOND_UNION"
  ],
  "IsNull": true,
  "Value": 0,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[15,17.5)",
    "US_NEVER_IN_UNION"
  ],
  "IsNull": false,
  "Value": 0.01831399787736417,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[15,17.5)",
    "US_FIRST_UNION_PERIOD1"
  ],
  "IsNull": false,
  "Value": 0.31524674743336684,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[15,17.5)",
    "US_FIRST_UNION_PERIOD2"
  ],
  "IsNull": true,
  "Value": 0,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[15,17.5)",
    "US_AFTER_FIRST_UNION"
  ],
  "IsNull": false,
  "Value": 0,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[15,17.5)",
    "US_SECOND_UNION"
  ],
  "IsNull": false,
  "Value": 0,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
```

```
{
  "Dims": [
    "[15,17.5)",
    "US_AFTER_SECOND_UNION"
  ],
  "IsNull": true,
  "Value": 0,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[17.5,20)",
    "US_NEVER_IN_UNION"
  ],
  "IsNull": false,
  "Value": 0.05375412945398035,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[17.5,20)",
    "US_FIRST_UNION_PERIOD1"
  ],
  "IsNull": false,
  "Value": 0.7131296479419359,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[17.5,20)",
    "US_FIRST_UNION_PERIOD2"
  ],
  "IsNull": false,
  "Value": 0.27913884088946966,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[17.5,20)",
    "US_AFTER_FIRST_UNION"
  ],
  "IsNull": false,
  "Value": 0.03403061129977049,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[17.5,20)",
    "US_SECOND_UNION"
  ],
  "IsNull": false,
  "Value": 0.6313567713962284,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[17.5,20)",
    "US_AFTER_SECOND_UNION"
  ],
  "IsNull": true,
  "Value": 0,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[20,22.5)",
    "US_NEVER_IN_UNION"
  ],
  "IsNull": false,
  "Value": 0.054454055397003744,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[20,22.5)",
    "US_FIRST_UNION_PERIOD1"
  ],
  "IsNull": false,
  "Value": 0.8142261153929924,
  "CalcName": "Expr0",
```

```
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[20,22.5)",
      "US_FIRST_UNION_PERIOD2"
    ],
    "IsNull": false,
    "Value": 0.22599976710820624,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[20,22.5)",
      "US_AFTER_FIRST_UNION"
    ],
    "IsNull": false,
    "Value": 0.02252894224790954,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[20,22.5)",
      "US_SECOND_UNION"
    ],
    "IsNull": false,
    "Value": 0.552801004995511,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[20,22.5)",
      "US_AFTER_SECOND_UNION"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[22.5,25)",
      "US_NEVER_IN_UNION"
    ],
    "IsNull": false,
    "Value": 0.053909930664369304,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[22.5,25)",
      "US_FIRST_UNION_PERIOD1"
    ],
    "IsNull": false,
    "Value": 0.8095822302661376,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[22.5,25)",
      "US_FIRST_UNION_PERIOD2"
    ],
    "IsNull": false,
    "Value": 0.19515570806104682,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[22.5,25)",
      "US_AFTER_FIRST_UNION"
    ],
    "IsNull": false,
    "Value": 0.04149012389398613,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[22.5,25)",
      "US_SECOND_UNION"
    ],
    "IsNull": false,
```

```
    "IsNull": false,
    "Value": 0.5994579474940404,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[22.5,25)",
      "US_AFTER_SECOND_UNION"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[25,27.5)",
      "US_NEVER_IN_UNION"
    ],
    "IsNull": false,
    "Value": 0.041598506770988926,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[25,27.5)",
      "US_FIRST_UNION_PERIOD1"
    ],
    "IsNull": false,
    "Value": 0.6024591444120154,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[25,27.5)",
      "US_FIRST_UNION_PERIOD2"
    ],
    "IsNull": false,
    "Value": 0.18708044915078834,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[25,27.5)",
      "US_AFTER_FIRST_UNION"
    ],
    "IsNull": false,
    "Value": 0.020349606948328054,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[25,27.5)",
      "US_SECOND_UNION"
    ],
    "IsNull": false,
    "Value": 1.0090672465971064,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[25,27.5)",
      "US_AFTER_SECOND_UNION"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[27.5,30)",
      "US_NEVER_IN_UNION"
    ],
    "IsNull": false,
    "Value": 0.03146736899461647,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[27.5,30)",
```

```
    "US_FIRST_UNION_PERIOD1"
  ],
  "IsNull": false,
  "Value": 0.4621989244428227,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[27.5,30)",
    "US_FIRST_UNION_PERIOD2"
  ],
  "IsNull": false,
  "Value": 0.1378516298224034,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[27.5,30)",
    "US_AFTER_FIRST_UNION"
  ],
  "IsNull": false,
  "Value": 0.022982490984116696,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[27.5,30)",
    "US_SECOND_UNION"
  ],
  "IsNull": false,
  "Value": 0.3622228137167969,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[27.5,30)",
    "US_AFTER_SECOND_UNION"
  ],
  "IsNull": false,
  "Value": 0.057202677626031914,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[30,32.5)",
    "US_NEVER_IN_UNION"
  ],
  "IsNull": false,
  "Value": 0.041649739660050096,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[30,32.5)",
    "US_FIRST_UNION_PERIOD1"
  ],
  "IsNull": false,
  "Value": 0.549536875782365,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[30,32.5)",
    "US_FIRST_UNION_PERIOD2"
  ],
  "IsNull": false,
  "Value": 0.1100888012811952,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[30,32.5)",
    "US_AFTER_FIRST_UNION"
  ],
  "IsNull": false,
  "Value": 0.04683269746871016,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
```

{
  "Dims": [
    "[30,32.5)",
    "US_SECOND_UNION"
  ],
  "IsNull": false,
  "Value": 0.33029219738801313,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[30,32.5)",
    "US_AFTER_SECOND_UNION"
  ],
  "IsNull": false,
  "Value": 0,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[32.5,35)",
    "US_NEVER_IN_UNION"
  ],
  "IsNull": false,
  "Value": 0.022147682759750396,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[32.5,35)",
    "US_FIRST_UNION_PERIOD1"
  ],
  "IsNull": false,
  "Value": 0.24011920351426697,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[32.5,35)",
    "US_FIRST_UNION_PERIOD2"
  ],
  "IsNull": false,
  "Value": 0.07322050511146887,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[32.5,35)",
    "US_AFTER_FIRST_UNION"
  ],
  "IsNull": false,
  "Value": 0,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[32.5,35)",
    "US_SECOND_UNION"
  ],
  "IsNull": false,
  "Value": 0.3916092867847636,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[32.5,35)",
    "US_AFTER_SECOND_UNION"
  ],
  "IsNull": false,
  "Value": 0,
  "CalcName": "Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[35,37.5)",
    "US_NEVER_IN_UNION"
  ],
  "IsNull": false,
  "Value": 0.011848889031140083,
  "CalcName": "Expr0",

```
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[35,37.5)",
      "US_FIRST_UNION_PERIOD1"
    ],
    "IsNull": false,
    "Value": 0.13186273037645685,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[35,37.5)",
      "US_FIRST_UNION_PERIOD2"
    ],
    "IsNull": false,
    "Value": 0.08309987275700664,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[35,37.5)",
      "US_AFTER_FIRST_UNION"
    ],
    "IsNull": false,
    "Value": 0.022104741738314906,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[35,37.5)",
      "US_SECOND_UNION"
    ],
    "IsNull": false,
    "Value": 0.27986398799940426,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[35,37.5)",
      "US_AFTER_SECOND_UNION"
    ],
    "IsNull": false,
    "Value": 0.11444275060810742,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[37.5,40)",
      "US_NEVER_IN_UNION"
    ],
    "IsNull": false,
    "Value": 0.017525081049451968,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[37.5,40)",
      "US_FIRST_UNION_PERIOD1"
    ],
    "IsNull": false,
    "Value": 0.1577477795548111,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[37.5,40)",
      "US_FIRST_UNION_PERIOD2"
    ],
    "IsNull": false,
    "Value": 0.05727119253571546,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[37.5,40)",
      "US_AFTER_FIRST_UNION"
    ],
    "IsNull": false,
```

```
    "IsNull": false,
    "Value": 0,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[37.5,40)",
      "US_SECOND_UNION"
    ],
    "IsNull": false,
    "Value": 0.16945519905993123,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[37.5,40)",
      "US_AFTER_SECOND_UNION"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[40,∞)",
      "US_NEVER_IN_UNION"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[40,∞)",
      "US_FIRST_UNION_PERIOD1"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[40,∞)",
      "US_FIRST_UNION_PERIOD2"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[40,∞)",
      "US_AFTER_FIRST_UNION"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[40,∞)",
      "US_SECOND_UNION"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[40,∞)",
      "US_AFTER_SECOND_UNION"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "(-∞,15)",
```

```
    "US_NEVER_IN_UNION"
  ],
  "IsNull": false,
  "Value": 0,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "(-∞,15)",
    "US_FIRST_UNION_PERIOD1"
  ],
  "IsNull": true,
  "Value": 0,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "(-∞,15)",
    "US_FIRST_UNION_PERIOD2"
  ],
  "IsNull": true,
  "Value": 0,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "(-∞,15)",
    "US_AFTER_FIRST_UNION"
  ],
  "IsNull": true,
  "Value": 0,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "(-∞,15)",
    "US_SECOND_UNION"
  ],
  "IsNull": true,
  "Value": 0,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "(-∞,15)",
    "US_AFTER_SECOND_UNION"
  ],
  "IsNull": true,
  "Value": 0,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[15,17.5)",
    "US_NEVER_IN_UNION"
  ],
  "IsNull": false,
  "Value": 0.000973984611518573,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[15,17.5)",
    "US_FIRST_UNION_PERIOD1"
  ],
  "IsNull": false,
  "Value": 0.024860412545759547,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[15,17.5)",
    "US_FIRST_UNION_PERIOD2"
  ],
  "IsNull": true,
  "Value": 0,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
```

```
    "Dims": [
      "[15,17.5)",
      "US_AFTER_FIRST_UNION"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "OM_SE_Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[15,17.5)",
      "US_SECOND_UNION"
    ],
    "IsNull": true,
    "Value": 0,
    "CalcName": "OM_SE_Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[15,17.5)",
      "US_AFTER_SECOND_UNION"
    ],
    "IsNull": true,
    "Value": 0,
    "CalcName": "OM_SE_Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[17.5,20)",
      "US_NEVER_IN_UNION"
    ],
    "IsNull": false,
    "Value": 0.0021489962375885207,
    "CalcName": "OM_SE_Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[17.5,20)",
      "US_FIRST_UNION_PERIOD1"
    ],
    "IsNull": false,
    "Value": 0.016185741408360488,
    "CalcName": "OM_SE_Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[17.5,20)",
      "US_FIRST_UNION_PERIOD2"
    ],
    "IsNull": false,
    "Value": 0.04582070104311915,
    "CalcName": "OM_SE_Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[17.5,20)",
      "US_AFTER_FIRST_UNION"
    ],
    "IsNull": false,
    "Value": 0.025795347742306313,
    "CalcName": "OM_SE_Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[17.5,20)",
      "US_SECOND_UNION"
    ],
    "IsNull": false,
    "Value": 0.7869023105120465,
    "CalcName": "OM_SE_Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[17.5,20)",
      "US_AFTER_SECOND_UNION"
    ],
    "IsNull": true,
    "Value": 0,
    "CalcName": "OM_SE_Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
```

"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[20,22.5)",
"US_NEVER_IN_UNION"
],
"IsNull": false,
"Value": 0.004127288570814012,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[20,22.5)",
"US_FIRST_UNION_PERIOD1"
],
"IsNull": false,
"Value": 0.02043658375123932,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[20,22.5)",
"US_FIRST_UNION_PERIOD2"
],
"IsNull": false,
"Value": 0.041408824916237076,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[20,22.5)",
"US_AFTER_FIRST_UNION"
],
"IsNull": false,
"Value": 0.012711616731356437,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[20,22.5)",
"US_SECOND_UNION"
],
"IsNull": false,
"Value": 0.17542003779519497,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[20,22.5)",
"US_AFTER_SECOND_UNION"
],
"IsNull": true,
"Value": 0,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[22.5,25)",
"US_NEVER_IN_UNION"
],
"IsNull": false,
"Value": 0.003881467816107038,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[22.5,25)",
"US_FIRST_UNION_PERIOD1"
],
"IsNull": false,
"Value": 0.016193285699936572,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[22.5,25)",
"US_FIRST_UNION_PERIOD2"
],
"IsNull": false,

"Value": 0.026391425572356776,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[22.5,25)",
"US_AFTER_FIRST_UNION"
],
"IsNull": false,
"Value": 0.024820325593176193,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[22.5,25)",
"US_SECOND_UNION"
],
"IsNull": false,
"Value": 0.3528132665325038,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[22.5,25)",
"US_AFTER_SECOND_UNION"
],
"IsNull": true,
"Value": 0,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[25,27.5)",
"US_NEVER_IN_UNION"
],
"IsNull": false,
"Value": 0.0036557290231329803,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[25,27.5)",
"US_FIRST_UNION_PERIOD1"
],
"IsNull": false,
"Value": 0.01959960089317563,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[25,27.5)",
"US_FIRST_UNION_PERIOD2"
],
"IsNull": false,
"Value": 0.02072720867372254,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[25,27.5)",
"US_AFTER_FIRST_UNION"
],
"IsNull": false,
"Value": 0.03151338138462493,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[25,27.5)",
"US_SECOND_UNION"
],
"IsNull": false,
"Value": 0.5935067944778238,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[25,27.5)",
"US_AFTER_SECOND_UNION"

```
    "US_AFTER_SECOND_UNION"
  ],
  "IsNull": false,
  "Value": 0,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[27.5,30)",
    "US_NEVER_IN_UNION"
  ],
  "IsNull": false,
  "Value": 0.005196619763242931,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[27.5,30)",
    "US_FIRST_UNION_PERIOD1"
  ],
  "IsNull": false,
  "Value": 0.011665634117207072,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[27.5,30)",
    "US_FIRST_UNION_PERIOD2"
  ],
  "IsNull": false,
  "Value": 0.01945905164366974,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[27.5,30)",
    "US_AFTER_FIRST_UNION"
  ],
  "IsNull": false,
  "Value": 0.029555729394609345,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[27.5,30)",
    "US_SECOND_UNION"
  ],
  "IsNull": false,
  "Value": 0.10422127860974977,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[27.5,30)",
    "US_AFTER_SECOND_UNION"
  ],
  "IsNull": false,
  "Value": 0.02670576575482527,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[30,32.5)",
    "US_NEVER_IN_UNION"
  ],
  "IsNull": false,
  "Value": 0.004163546631457847,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[30,32.5)",
    "US_FIRST_UNION_PERIOD1"
  ],
  "IsNull": false,
  "Value": 0.07776769082434758,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
```

```
    "Dims": [
      "[30,32.5)",
      "US_FIRST_UNION_PERIOD2"
    ],
    "IsNull": false,
    "Value": 0.00807157824867373,
    "CalcName": "OM_SE_Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[30,32.5)",
      "US_AFTER_FIRST_UNION"
    ],
    "IsNull": false,
    "Value": 0.04520648331822375,
    "CalcName": "OM_SE_Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[30,32.5)",
      "US_SECOND_UNION"
    ],
    "IsNull": false,
    "Value": 0.23240410935784928,
    "CalcName": "OM_SE_Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[30,32.5)",
      "US_AFTER_SECOND_UNION"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "OM_SE_Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[32.5,35)",
      "US_NEVER_IN_UNION"
    ],
    "IsNull": false,
    "Value": 0.002133413207162846,
    "CalcName": "OM_SE_Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[32.5,35)",
      "US_FIRST_UNION_PERIOD1"
    ],
    "IsNull": false,
    "Value": 0.0522818322110285,
    "CalcName": "OM_SE_Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[32.5,35)",
      "US_FIRST_UNION_PERIOD2"
    ],
    "IsNull": false,
    "Value": 0.01949087594847643,
    "CalcName": "OM_SE_Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[32.5,35)",
      "US_AFTER_FIRST_UNION"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "OM_SE_Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[32.5,35)",
      "US_SECOND_UNION"
    ],
    "IsNull": false,
    "Value": 0.4087021371108991,
    "CalcName": "OM_SE_Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
```

```
},
{
  "Dims": [
    "[32.5,35)",
    "US_AFTER_SECOND_UNION"
  ],
  "IsNull": false,
  "Value": 0,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[35,37.5)",
    "US_NEVER_IN_UNION"
  ],
  "IsNull": false,
  "Value": 0.004685194405920375,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[35,37.5)",
    "US_FIRST_UNION_PERIOD1"
  ],
  "IsNull": false,
  "Value": 0.03917868926846236,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[35,37.5)",
    "US_FIRST_UNION_PERIOD2"
  ],
  "IsNull": false,
  "Value": 0.012717967350323902,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[35,37.5)",
    "US_AFTER_FIRST_UNION"
  ],
  "IsNull": false,
  "Value": 0.012339858333394415,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[35,37.5)",
    "US_SECOND_UNION"
  ],
  "IsNull": false,
  "Value": 0.14037654439621916,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[35,37.5)",
    "US_AFTER_SECOND_UNION"
  ],
  "IsNull": false,
  "Value": 0.05343598681843709,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[37.5,40)",
    "US_NEVER_IN_UNION"
  ],
  "IsNull": false,
  "Value": 0.003295964388970783,
  "CalcName": "OM_SE_Expr0",
  "RunDigest": "b794d3399099035740e117378c523feb"
},
{
  "Dims": [
    "[37.5,40)",
    "US_FIRST_UNION_PERIOD1"
  ],
  "IsNull": false,
  "Value": 0.019743934000037976,
```

"Value": 0.01974393400903797 6,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[37.5,40)",
"US_FIRST_UNION_PERIOD2"
],
"IsNull": false,
"Value": 0.01043050794857122,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[37.5,40)",
"US_AFTER_FIRST_UNION"
],
"IsNull": false,
"Value": 0,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[37.5,40)",
"US_SECOND_UNION"
],
"IsNull": false,
"Value": 0.11795500304906231,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[37.5,40)",
"US_AFTER_SECOND_UNION"
],
"IsNull": false,
"Value": 0,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[40,∞)",
"US_NEVER_IN_UNION"
],
"IsNull": false,
"Value": 0,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[40,∞)",
"US_FIRST_UNION_PERIOD1"
],
"IsNull": false,
"Value": 0,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[40,∞)",
"US_FIRST_UNION_PERIOD2"
],
"IsNull": false,
"Value": 0,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[40,∞)",
"US_AFTER_FIRST_UNION"
],
"IsNull": false,
"Value": 0,
"CalcName": "OM_SE_Expr0",
"RunDigest": "b794d3399099035740e117378c523feb"
},
{
"Dims": [
"[40,∞)",
"US_SECOND_UNION"

```
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "OM_SE_Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  },
  {
    "Dims": [
      "[40,∞)",
      "US_AFTER_SECOND_UNION"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "OM_SE_Expr0",
    "RunDigest": "b794d3399099035740e117378c523feb"
  }
]
```

GET output table values and compare model runs

Read a "page" of output table values and compare model runs.

Comparison can be calculated as one of the following:

- for each table expression use one of: `diff`, `ratio` or `percent` comparison between `[base]` and `[variant]` model runs.
- use comma separated list of comparison expressions between `[base]` and `[variant]` or simple expression for each run.

It is also possible to use parameter(s) in calculation, parameter must be a scalar of float or integer type. For example: `(Expr1[variant] - Expr1[base]) * param.Scale[base]`, where `param.Scale` is a value of scalar parameter `Scale` in `[base]` 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/compare/:compare/variant/:variant
GET /api/model/:model/run/:run/table/:name/compare/:compare/variant/:variant/start/:start
GET /api/model/:model/run/:run/table/:name/compare/:compare/variant/:variant/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) base model run digest, run stamp or run name
:variant - (required) variant model run(s): comma-separated list of digests, run stamps or run names
```

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
```

```
:compare - (required) comparison to calculate
```

- `diff` differenece of values between variant and base run, e.g.: `Expr0[variant] - Expr0[base]`
- `ratio` ratio of values between variant and base run, e.g.: `Expr0[variant] / Expr0[base]`
- `percent` proportional difference multipled by 100, e.g.: `100 * (Expr0[variant] - Expr0[base]) / Expr0[base]`

Or a list of comma-separated expressions, for example: `expr0, expr1[variant] + expr2[base]`

```
:start - (optional) start "page" row number, zero-based.
:count - (optional) "page" size, number of rows to select, if count <= 0 then all rows selected.
```

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/compare/diff/variant/Default-4
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/compare/ratio/variant/Default-4
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/compare/percent/variant/Default-4

http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/compare/ratio/variant/Default-4/start/2
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/compare/ratio/variant/Default-4/start/2/count/4

http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/compare/expr0%2Cexpr1%5Bvariant%5D%2Bexpr2%5Bbase%5D/variant/Default-4
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/compare/%28expr1%5Bvariant%5D-expr1%5Bbase%5D%29%2Bparam.StartingSeed%5Bbase%5D/variant/
Default-4
```

Note:

- `expr0%2Cexpr1%5Bvariant%5D%2Bexpr2%5Bbase%5D` is URL encoded: `expr0,expr1[variant]+expr2[base]`.
- `%28expr1%5Bvariant%5D-expr1%5Bbase%5D%29%2Bparam.StartingSeed%5Bbase%5D` is URL encoded: `(expr1[variant]-expr1[base])+param.StartingSeed[base]`.

Example:

```
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/compare/percent/variant/Default-4,Sub-values_4
```


```
[
  {
    "Dims": [
      "L",
      "M"
    ],
    "IsNull": false,
    "Value": 50,
    "CalcName": "expr0",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "L",
      "F"
    ],
    "IsNull": false,
    "Value": 60,
    "CalcName": "expr0",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "L",
      "all"
    ],
    "IsNull": false,
    "Value": 1,
    "CalcName": "expr0",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "M",
      "M"
    ],
    "IsNull": false,
    "Value": 51.599999999999994,
    "CalcName": "expr0",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "M",
      "F"
    ],
    "IsNull": false,
    "Value": 62,
    "CalcName": "expr0",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "M",
      "all"
    ],
    "IsNull": false,
    "Value": 2,
    "CalcName": "expr0",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "H",
      "M"
    ],
    "IsNull": false,
    "Value": 53.2,
    "CalcName": "expr0",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
```

```
      "H",
      "F"
    ],
    "IsNull": false,
    "Value": 64,
    "CalcName": "expr0",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "H",
      "all"
    ],
    "IsNull": false,
    "Value": 3,
    "CalcName": "expr0",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "L",
      "M"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "expr1",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "L",
      "F"
    ],
    "IsNull": false,
    "Value": 1,
    "CalcName": "expr1",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "L",
      "all"
    ],
    "IsNull": false,
    "Value": 800,
    "CalcName": "expr1",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "M",
      "M"
    ],
    "IsNull": false,
    "Value": 1,
    "CalcName": "expr1",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "M",
      "F"
    ],
    "IsNull": false,
    "Value": 2,
    "CalcName": "expr1",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "M",
      "all"
    ],
    "IsNull": false,
    "Value": 801,
    "CalcName": "expr1",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "H",
      "M"
    ],
    "IsNull": false,
    "Value": 2,
    "CalcName": "expr1",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
```

```
{
  "Dims": [
    "H",
    "F"
  ],
  "IsNull": false,
  "Value": 3,
  "CalcName": "expr1",
  "RunDigest": "ca663651953bae94d0afdf71edba4c91"
},
{
  "Dims": [
    "H",
    "all"
  ],
  "IsNull": false,
  "Value": 802,
  "CalcName": "expr1",
  "RunDigest": "ca663651953bae94d0afdf71edba4c91"
},
{
  "Dims": [
    "L",
    "M"
  ],
  "IsNull": false,
  "Value": 50,
  "CalcName": "expr2",
  "RunDigest": "ca663651953bae94d0afdf71edba4c91"
},
{
  "Dims": [
    "L",
    "F"
  ],
  "IsNull": false,
  "Value": 61,
  "CalcName": "expr2",
  "RunDigest": "ca663651953bae94d0afdf71edba4c91"
},
{
  "Dims": [
    "L",
    "all"
  ],
  "IsNull": false,
  "Value": 801,
  "CalcName": "expr2",
  "RunDigest": "ca663651953bae94d0afdf71edba4c91"
},
{
  "Dims": [
    "M",
    "M"
  ],
  "IsNull": false,
  "Value": 52.599999999999994,
  "CalcName": "expr2",
  "RunDigest": "ca663651953bae94d0afdf71edba4c91"
},
{
  "Dims": [
    "M",
    "F"
  ],
  "IsNull": false,
  "Value": 64,
  "CalcName": "expr2",
  "RunDigest": "ca663651953bae94d0afdf71edba4c91"
},
{
  "Dims": [
    "M",
    "all"
  ],
  "IsNull": false,
  "Value": 803,
  "CalcName": "expr2",
  "RunDigest": "ca663651953bae94d0afdf71edba4c91"
},
{
  "Dims": [
    "H",
    "M"
  ],
  "IsNull": false,
  "Value": 55.2,
  "CalcName": "expr2",
```

```
    "CalcName": "expr2",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "H",
      "F"
    ],
    "IsNull": false,
    "Value": 67,
    "CalcName": "expr2",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "H",
      "all"
    ],
    "IsNull": false,
    "Value": 805,
    "CalcName": "expr2",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "L",
      "M"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "expr3",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "L",
      "F"
    ],
    "IsNull": false,
    "Value": 60,
    "CalcName": "expr3",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "L",
      "all"
    ],
    "IsNull": false,
    "Value": 800,
    "CalcName": "expr3",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "M",
      "M"
    ],
    "IsNull": false,
    "Value": 51.599999999999994,
    "CalcName": "expr3",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "M",
      "F"
    ],
    "IsNull": false,
    "Value": 124,
    "CalcName": "expr3",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "M",
      "all"
    ],
    "IsNull": false,
    "Value": 1602,
    "CalcName": "expr3",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "H",
      "M"
    ],
```

```
    "IsNull": false,
    "Value": 106.4,
    "CalcName": "expr3",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "H",
      "F"
    ],
    "IsNull": false,
    "Value": 192,
    "CalcName": "expr3",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "H",
      "all"
    ],
    "IsNull": false,
    "Value": 2406,
    "CalcName": "expr3",
    "RunDigest": "ca663651953bae94d0afdf71edba4c91"
  },
  {
    "Dims": [
      "L",
      "M"
    ],
    "IsNull": false,
    "Value": 50,
    "CalcName": "expr0",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "L",
      "F"
    ],
    "IsNull": false,
    "Value": 60,
    "CalcName": "expr0",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "L",
      "all"
    ],
    "IsNull": false,
    "Value": 1201,
    "CalcName": "expr0",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "M",
      "M"
    ],
    "IsNull": false,
    "Value": 51.599999999999994,
    "CalcName": "expr0",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "M",
      "F"
    ],
    "IsNull": false,
    "Value": 62,
    "CalcName": "expr0",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "M",
      "all"
    ],
    "IsNull": false,
    "Value": 1202,
    "CalcName": "expr0",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "H",
```

```
    "M"
  ],
  "IsNull": false,
  "Value": 53.2,
  "CalcName": "expr0",
  "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
},
{
  "Dims": [
    "H",
    "F"
  ],
  "IsNull": false,
  "Value": 64,
  "CalcName": "expr0",
  "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
},
{
  "Dims": [
    "H",
    "all"
  ],
  "IsNull": false,
  "Value": 1203,
  "CalcName": "expr0",
  "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
},
{
  "Dims": [
    "L",
    "M"
  ],
  "IsNull": false,
  "Value": 6,
  "CalcName": "expr1",
  "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
},
{
  "Dims": [
    "L",
    "F"
  ],
  "IsNull": false,
  "Value": 10,
  "CalcName": "expr1",
  "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
},
{
  "Dims": [
    "L",
    "all"
  ],
  "IsNull": false,
  "Value": 3206,
  "CalcName": "expr1",
  "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
},
{
  "Dims": [
    "M",
    "M"
  ],
  "IsNull": false,
  "Value": 10,
  "CalcName": "expr1",
  "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
},
{
  "Dims": [
    "M",
    "F"
  ],
  "IsNull": false,
  "Value": 14,
  "CalcName": "expr1",
  "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
},
{
  "Dims": [
    "M",
    "all"
  ],
  "IsNull": false,
  "Value": 3210,
  "CalcName": "expr1",
  "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
},
```

```
{
  "Dims": [
    "H",
    "M"
  ],
  "IsNull": false,
  "Value": 14,
  "CalcName": "expr1",
  "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
},
{
  "Dims": [
    "H",
    "F"
  ],
  "IsNull": false,
  "Value": 18,
  "CalcName": "expr1",
  "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
},
{
  "Dims": [
    "H",
    "all"
  ],
  "IsNull": false,
  "Value": 3214,
  "CalcName": "expr1",
  "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
},
{
  "Dims": [
    "L",
    "M"
  ],
  "IsNull": false,
  "Value": 51.5,
  "CalcName": "expr2",
  "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
},
{
  "Dims": [
    "L",
    "F"
  ],
  "IsNull": false,
  "Value": 62.5,
  "CalcName": "expr2",
  "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
},
{
  "Dims": [
    "L",
    "all"
  ],
  "IsNull": false,
  "Value": 2002.5,
  "CalcName": "expr2",
  "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
},
{
  "Dims": [
    "M",
    "M"
  ],
  "IsNull": false,
  "Value": 54.099999999999994,
  "CalcName": "expr2",
  "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
},
{
  "Dims": [
    "M",
    "F"
  ],
  "IsNull": false,
  "Value": 65.5,
  "CalcName": "expr2",
  "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
},
{
  "Dims": [
    "M",
    "all"
  ],
  "IsNull": false,
  "Value": 2004.5,
  "CalcName": "expr2",
```

```
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "H",
      "M"
    ],
    "IsNull": false,
    "Value": 56.7,
    "CalcName": "expr2",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "H",
      "F"
    ],
    "IsNull": false,
    "Value": 68.5,
    "CalcName": "expr2",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "H",
      "all"
    ],
    "IsNull": false,
    "Value": 2006.5,
    "CalcName": "expr2",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "L",
      "M"
    ],
    "IsNull": false,
    "Value": 75,
    "CalcName": "expr3",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "L",
      "F"
    ],
    "IsNull": false,
    "Value": 150,
    "CalcName": "expr3",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "L",
      "all"
    ],
    "IsNull": false,
    "Value": 963601.5,
    "CalcName": "expr3",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "M",
      "M"
    ],
    "IsNull": false,
    "Value": 129,
    "CalcName": "expr3",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "M",
      "F"
    ],
    "IsNull": false,
    "Value": 217,
    "CalcName": "expr3",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "M",
      "all"
    ],
    "IsNull": false,
```

```
    "IsNull": false,
    "Value": 965605,
    "CalcName": "expr3",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "H",
      "M"
    ],
    "IsNull": false,
    "Value": 186.2,
    "CalcName": "expr3",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "H",
      "F"
    ],
    "IsNull": false,
    "Value": 288,
    "CalcName": "expr3",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "H",
      "all"
    ],
    "IsNull": false,
    "Value": 967610.5,
    "CalcName": "expr3",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "L",
      "M"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "percent_expr0",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "L",
      "F"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "percent_expr0",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "L",
      "all"
    ],
    "IsNull": false,
    "Value": 1200,
    "CalcName": "percent_expr0",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "M",
      "M"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "percent_expr0",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "M",
      "F"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "percent_expr0",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "M",
```

```
      "all"
    ],
    "IsNull": false,
    "Value": 60000,
    "CalcName": "percent_expr0",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "H",
      "M"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "percent_expr0",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "H",
      "F"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "percent_expr0",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "H",
      "all"
    ],
    "IsNull": false,
    "Value": 40000,
    "CalcName": "percent_expr0",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "L",
      "M"
    ],
    "IsNull": true,
    "Value": 0,
    "CalcName": "percent_expr1",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "L",
      "F"
    ],
    "IsNull": false,
    "Value": 900,
    "CalcName": "percent_expr1",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "L",
      "all"
    ],
    "IsNull": false,
    "Value": 300.75,
    "CalcName": "percent_expr1",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "M",
      "M"
    ],
    "IsNull": false,
    "Value": 900,
    "CalcName": "percent_expr1",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "M",
      "F"
    ],
    "IsNull": false,
    "Value": 600,
    "CalcName": "percent_expr1",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
```

{
  "Dims": [
    "M",
    "all"
  ],
  "IsNull": false,
  "Value": 300.749063670412,
  "CalcName": "percent_expr1",
  "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
},
{
  "Dims": [
    "H",
    "M"
  ],
  "IsNull": false,
  "Value": 600,
  "CalcName": "percent_expr1",
  "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
},
{
  "Dims": [
    "H",
    "F"
  ],
  "IsNull": false,
  "Value": 500,
  "CalcName": "percent_expr1",
  "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
},
{
  "Dims": [
    "H",
    "all"
  ],
  "IsNull": false,
  "Value": 300.74812967581045,
  "CalcName": "percent_expr1",
  "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
},
{
  "Dims": [
    "L",
    "M"
  ],
  "IsNull": false,
  "Value": 3,
  "CalcName": "percent_expr2",
  "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
},
{
  "Dims": [
    "L",
    "F"
  ],
  "IsNull": false,
  "Value": 2.459016393442623,
  "CalcName": "percent_expr2",
  "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
},
{
  "Dims": [
    "L",
    "all"
  ],
  "IsNull": false,
  "Value": 150,
  "CalcName": "percent_expr2",
  "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
},
{
  "Dims": [
    "M",
    "M"
  ],
  "IsNull": false,
  "Value": 2.8517110266159698,
  "CalcName": "percent_expr2",
  "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
},
{
  "Dims": [
    "M",
    "F"
  ],
  "IsNull": false,
  "Value": 2.34375,
  "CalcName": "percent_expr2",

```
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "M",
      "all"
    ],
    "IsNull": false,
    "Value": 149.626400996264,
    "CalcName": "percent_expr2",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "H",
      "M"
    ],
    "IsNull": false,
    "Value": 2.717391304347826,
    "CalcName": "percent_expr2",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "H",
      "F"
    ],
    "IsNull": false,
    "Value": 2.2388059701492535,
    "CalcName": "percent_expr2",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "H",
      "all"
    ],
    "IsNull": false,
    "Value": 149.25465838509317,
    "CalcName": "percent_expr2",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "L",
      "M"
    ],
    "IsNull": true,
    "Value": 0,
    "CalcName": "percent_expr3",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "L",
      "F"
    ],
    "IsNull": false,
    "Value": 150,
    "CalcName": "percent_expr3",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "L",
      "all"
    ],
    "IsNull": false,
    "Value": 120350.1875,
    "CalcName": "percent_expr3",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "M",
      "M"
    ],
    "IsNull": false,
    "Value": 150.00000000000003,
    "CalcName": "percent_expr3",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "M",
      "F"
    ],
    "IsNull": false,
```

```
    "Value": 75,
    "CalcName": "percent_expr3",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "M",
      "all"
    ],
    "IsNull": false,
    "Value": 60174.968789013736,
    "CalcName": "percent_expr3",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "H",
      "M"
    ],
    "IsNull": false,
    "Value": 74.99999999999999,
    "CalcName": "percent_expr3",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "H",
      "F"
    ],
    "IsNull": false,
    "Value": 50,
    "CalcName": "percent_expr3",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "H",
      "all"
    ],
    "IsNull": false,
    "Value": 40116.56275976725,
    "CalcName": "percent_expr3",
    "RunDigest": "c519fc5869f244ac4c80ae44695a4272"
  },
  {
    "Dims": [
      "L",
      "M"
    ],
    "IsNull": false,
    "Value": 50,
    "CalcName": "expr0",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "L",
      "F"
    ],
    "IsNull": false,
    "Value": 60,
    "CalcName": "expr0",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "L",
      "all"
    ],
    "IsNull": false,
    "Value": 1201,
    "CalcName": "expr0",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "M",
      "M"
    ],
    "IsNull": false,
    "Value": 51.599999999999994,
    "CalcName": "expr0",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "M",
      "F"
```

```
    "F"
  ],
  "IsNull": false,
  "Value": 62,
  "CalcName": "expr0",
  "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
},
{
  "Dims": [
    "M",
    "all"
  ],
  "IsNull": false,
  "Value": 1202,
  "CalcName": "expr0",
  "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
},
{
  "Dims": [
    "H",
    "M"
  ],
  "IsNull": false,
  "Value": 53.2,
  "CalcName": "expr0",
  "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
},
{
  "Dims": [
    "H",
    "F"
  ],
  "IsNull": false,
  "Value": 64,
  "CalcName": "expr0",
  "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
},
{
  "Dims": [
    "H",
    "all"
  ],
  "IsNull": false,
  "Value": 1203,
  "CalcName": "expr0",
  "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
},
{
  "Dims": [
    "L",
    "M"
  ],
  "IsNull": false,
  "Value": 6,
  "CalcName": "expr1",
  "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
},
{
  "Dims": [
    "L",
    "F"
  ],
  "IsNull": false,
  "Value": 10,
  "CalcName": "expr1",
  "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
},
{
  "Dims": [
    "L",
    "all"
  ],
  "IsNull": false,
  "Value": 3206,
  "CalcName": "expr1",
  "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
},
{
  "Dims": [
    "M",
    "M"
  ],
  "IsNull": false,
  "Value": 10,
  "CalcName": "expr1",
  "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
},
{
```

```
    "Dims": [
      "M",
      "F"
    ],
    "IsNull": false,
    "Value": 14,
    "CalcName": "expr1",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "M",
      "all"
    ],
    "IsNull": false,
    "Value": 3210,
    "CalcName": "expr1",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "H",
      "M"
    ],
    "IsNull": false,
    "Value": 14,
    "CalcName": "expr1",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "H",
      "F"
    ],
    "IsNull": false,
    "Value": 18,
    "CalcName": "expr1",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "H",
      "all"
    ],
    "IsNull": false,
    "Value": 3214,
    "CalcName": "expr1",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "L",
      "M"
    ],
    "IsNull": false,
    "Value": 51.5,
    "CalcName": "expr2",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "L",
      "F"
    ],
    "IsNull": false,
    "Value": 62.5,
    "CalcName": "expr2",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "L",
      "all"
    ],
    "IsNull": false,
    "Value": 2002.5,
    "CalcName": "expr2",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "M",
      "M"
    ],
    "IsNull": false,
    "Value": 54.099999999999994,
    "CalcName": "expr2",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
```

```
},
{
  "Dims": [
    "M",
    "F"
  ],
  "IsNull": false,
  "Value": 65.5,
  "CalcName": "expr2",
  "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
},
{
  "Dims": [
    "M",
    "all"
  ],
  "IsNull": false,
  "Value": 2004.5,
  "CalcName": "expr2",
  "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
},
{
  "Dims": [
    "H",
    "M"
  ],
  "IsNull": false,
  "Value": 56.7,
  "CalcName": "expr2",
  "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
},
{
  "Dims": [
    "H",
    "F"
  ],
  "IsNull": false,
  "Value": 68.5,
  "CalcName": "expr2",
  "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
},
{
  "Dims": [
    "H",
    "all"
  ],
  "IsNull": false,
  "Value": 2006.5,
  "CalcName": "expr2",
  "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
},
{
  "Dims": [
    "L",
    "M"
  ],
  "IsNull": false,
  "Value": 75,
  "CalcName": "expr3",
  "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
},
{
  "Dims": [
    "L",
    "F"
  ],
  "IsNull": false,
  "Value": 150,
  "CalcName": "expr3",
  "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
},
{
  "Dims": [
    "L",
    "all"
  ],
  "IsNull": false,
  "Value": 963601.5,
  "CalcName": "expr3",
  "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
},
{
  "Dims": [
    "M",
    "M"
  ],
  "IsNull": false,
```

```
    "Value": 129,
    "CalcName": "expr3",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "M",
      "F"
    ],
    "IsNull": false,
    "Value": 217,
    "CalcName": "expr3",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "M",
      "all"
    ],
    "IsNull": false,
    "Value": 965605,
    "CalcName": "expr3",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "H",
      "M"
    ],
    "IsNull": false,
    "Value": 186.2,
    "CalcName": "expr3",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "H",
      "F"
    ],
    "IsNull": false,
    "Value": 288,
    "CalcName": "expr3",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "H",
      "all"
    ],
    "IsNull": false,
    "Value": 967610.5,
    "CalcName": "expr3",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "L",
      "M"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "percent_expr0",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "L",
      "F"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "percent_expr0",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "L",
      "all"
    ],
    "IsNull": false,
    "Value": 1200,
    "CalcName": "percent_expr0",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "M",
      "M"
```

```
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "percent_expr0",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "M",
      "F"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "percent_expr0",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "M",
      "all"
    ],
    "IsNull": false,
    "Value": 60000,
    "CalcName": "percent_expr0",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "H",
      "M"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "percent_expr0",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "H",
      "F"
    ],
    "IsNull": false,
    "Value": 0,
    "CalcName": "percent_expr0",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "H",
      "all"
    ],
    "IsNull": false,
    "Value": 40000,
    "CalcName": "percent_expr0",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "L",
      "M"
    ],
    "IsNull": true,
    "Value": 0,
    "CalcName": "percent_expr1",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "L",
      "F"
    ],
    "IsNull": false,
    "Value": 900,
    "CalcName": "percent_expr1",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "L",
      "all"
    ],
    "IsNull": false,
    "Value": 300.75,
    "CalcName": "percent_expr1",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
```

```
    "Dims": [
      "M",
      "M"
    ],
    "IsNull": false,
    "Value": 900,
    "CalcName": "percent_expr1",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "M",
      "F"
    ],
    "IsNull": false,
    "Value": 600,
    "CalcName": "percent_expr1",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "M",
      "all"
    ],
    "IsNull": false,
    "Value": 300.749063670412,
    "CalcName": "percent_expr1",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "H",
      "M"
    ],
    "IsNull": false,
    "Value": 600,
    "CalcName": "percent_expr1",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "H",
      "F"
    ],
    "IsNull": false,
    "Value": 500,
    "CalcName": "percent_expr1",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "H",
      "all"
    ],
    "IsNull": false,
    "Value": 300.74812967581045,
    "CalcName": "percent_expr1",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "L",
      "M"
    ],
    "IsNull": false,
    "Value": 3,
    "CalcName": "percent_expr2",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "L",
      "F"
    ],
    "IsNull": false,
    "Value": 2.459016393442623,
    "CalcName": "percent_expr2",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "L",
      "all"
    ],
    "IsNull": false,
    "Value": 150,
    "CalcName": "percent_expr2",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
```

```
  },
  {
    "Dims": [
      "M",
      "M"
    ],
    "IsNull": false,
    "Value": 2.8517110266159698,
    "CalcName": "percent_expr2",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "M",
      "F"
    ],
    "IsNull": false,
    "Value": 2.34375,
    "CalcName": "percent_expr2",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "M",
      "all"
    ],
    "IsNull": false,
    "Value": 149.626400996264,
    "CalcName": "percent_expr2",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "H",
      "M"
    ],
    "IsNull": false,
    "Value": 2.717391304347826,
    "CalcName": "percent_expr2",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "H",
      "F"
    ],
    "IsNull": false,
    "Value": 2.2388059701492535,
    "CalcName": "percent_expr2",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "H",
      "all"
    ],
    "IsNull": false,
    "Value": 149.25465838509317,
    "CalcName": "percent_expr2",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "L",
      "M"
    ],
    "IsNull": true,
    "Value": 0,
    "CalcName": "percent_expr3",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "L",
      "F"
    ],
    "IsNull": false,
    "Value": 150,
    "CalcName": "percent_expr3",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "L",
      "all"
    ],
    "IsNull": false,
    "Value": 120350.1875,
```

```
    "CalcName": "percent_expr3",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "M",
      "M"
    ],
    "IsNull": false,
    "Value": 150.00000000000003,
    "CalcName": "percent_expr3",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "M",
      "F"
    ],
    "IsNull": false,
    "Value": 75,
    "CalcName": "percent_expr3",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "M",
      "all"
    ],
    "IsNull": false,
    "Value": 60174.968789013736,
    "CalcName": "percent_expr3",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "H",
      "M"
    ],
    "IsNull": false,
    "Value": 74.99999999999999,
    "CalcName": "percent_expr3",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "H",
      "F"
    ],
    "IsNull": false,
    "Value": 50,
    "CalcName": "percent_expr3",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  },
  {
    "Dims": [
      "H",
      "all"
    ],
    "IsNull": false,
    "Value": 40116.56275976725,
    "CalcName": "percent_expr3",
    "RunDigest": "ffad6e8ed4449dafe11b82c7aea82f74"
  }
]
```

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, if count <= 0 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, if count <= 0 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, if count <= 0 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,
        "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,
```

```
            "IsNull": false,
            "Value": "Part"
        }, {
            "IsNull": false,
            "Value": "false"
        }, {
            "IsNull": false,
            "Value": "0"
        }
    ]
  }
]
```

GET aggregated microdata from model run

Read a "page" of aggregated microdata values from model run.

Result can include multiple aggregations of value attributes (float of integer type) and group by dimension attributes (enum-based or bool type). Aggregation(s) is a comma-separated list of Model Output Expressions of microdata value attributes. For example, two aggregations: `OM_AVG(Income) , OM_MAX(Salary + Pension)` and group by two dimension attributes: `AgeGroup , Sex`.

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.

It is also possible to use parameter(s) in calculation, parameter must be a scalar of float or integer type. For example: `OM_COUNT_IF(Income > param.High)`, where `param.High` is a value of scalar parameter `High` in that model run.

Enum-based microdata attributes returned as enum codes.

Following aggregation functions avaliable:

- `OM_AVG` mean of accumulators sub-values
- `OM_SUM` sum of accumulators sub-values
- `OM_COUNT` count of accumulators sub-values (excluding NULL's)
- `OM_COUNT_IF` count values matching condition
- `OM_MAX` maximum of accumulators sub-values
- `OM_MIN` minimum of accumulators sub-values
- `OM_VAR` variance of accumulators sub-values
- `OM_SD` standard deviation of accumulators sub-values
- `OM_SE` standard error of accumulators sub-values
- `OM_CV` coefficient of variation of accumulators sub-values

For more details please see: Model Output Expressions

Methods:

```
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/calc/:calc
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/calc/:calc/start/:start
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/calc/:calc/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
```

```
:group-by - (required) comma-separated list of dimension attribute(s) to group by aggreagated values, dimension attribute must be enum-based or boolean type.
```

```
:calc - (required) comma-separated list of aggregation of microdata value attribute(s), value attribute must be float of integer type.
```

```
:start - (optional) start "page" row number, zero-based.
:count - (optional) "page" size, number of rows to select, if count <= 0 then all rows selected.
```

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/calc/OM_AVG(Income)
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/calc/OM_AVG(Income)/start/2
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/calc/OM_AVG(Income)/start/2/count/3

http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/calc/OM_AVG(Income),OM_AVG(Salary+Pension)
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/calc/OM_COUNT_IF(Income%3Eparam.StartingSeed)
```

Note: `OM_COUNT_IF(Income%3Eparam.StartingSeed)` is URL encoded: `OM_COUNT_IF(Income>param.StartingSeed)`.

Return example:

```
curl http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/calc/OM_AVG(Income)/csv
```


```
[
  {
    "Attr": [
      {
        "IsNull": false,
        "Value": "10-20"
      },
      {
        "IsNull": false,
        "Value": "M"
      },
      {
        "IsNull": false,
        "Value": 13400876.354360776
      }
    ],
    "CalcName": "ex_12000",
    "RunDigest": "a59c91359c4cd98f6275529c798d2485"
  },
  {
    "Attr": [
      {
        "IsNull": false,
        "Value": "10-20"
      },
      {
        "IsNull": false,
        "Value": "F"
      },
      {
        "IsNull": false,
        "Value": 13403741.889092576
      }
    ],
    "CalcName": "ex_12000",
    "RunDigest": "a59c91359c4cd98f6275529c798d2485"
  },
  {
    "Attr": [
      {
        "IsNull": false,
        "Value": "20-30"
      },
      {
        "IsNull": false,
        "Value": "M"
      },
      {
        "IsNull": false,
        "Value": 134201344
      }
    ],
    "CalcName": "ex_12000",
    "RunDigest": "a59c91359c4cd98f6275529c798d2485"
  },
  {
    "Attr": [
      {
        "IsNull": false,
```

```
        "Value": "20-30"
      },
      {
        "IsNull": false,
        "Value": "F"
      },
      {
        "IsNull": false,
        "Value": 134209535
      }
    ],
    "CalcName": "ex_12000",
    "RunDigest": "a59c91359c4cd98f6275529c798d2485"
  },
  {
    "Attr": [
      {
        "IsNull": false,
        "Value": "30-40"
      },
      {
        "IsNull": false,
        "Value": "M"
      },
      {
        "IsNull": false,
        "Value": 134283254
      }
    ],
    "CalcName": "ex_12000",
    "RunDigest": "a59c91359c4cd98f6275529c798d2485"
  },
  {
    "Attr": [
      {
        "IsNull": false,
        "Value": "30-40"
      },
      {
        "IsNull": false,
        "Value": "F"
      },
      {
        "IsNull": false,
        "Value": 134291445
      }
    ],
    "CalcName": "ex_12000",
    "RunDigest": "a59c91359c4cd98f6275529c798d2485"
  },
  {
    "Attr": [
      {
        "IsNull": false,
        "Value": "40+"
      },
      {
        "IsNull": false,
        "Value": "M"
      },
      {
        "IsNull": false,
        "Value": 74645804.26116003
      }
    ],
    "CalcName": "ex_12000",
    "RunDigest": "a59c91359c4cd98f6275529c798d2485"
  },
  {
    "Attr": [
      {
        "IsNull": false,
        "Value": "40+"
      },
      {
        "IsNull": false,
        "Value": "F"
      },
      {
        "IsNull": false,
        "Value": 71069306.57187325
      }
    ],
    "CalcName": "ex_12000",
    "RunDigest": "a59c91359c4cd98f6275529c798d2485"
  }
]
```

GET microdata run comparison

Read a "page" of microdata values and compare model runs.

Compare `[base]` and `[variant]` model runs microdata value attributes (float of integer type), group it by dimension attributes (enum-based or bool type).

Result can include multiple aggregated comparisons, grouped by multiple dimension attributes. Aggregated comparision(s) is a comma-separated list of Model Output Expressions of `[base]` and `[variant]` value attributes. For example, two comparisions: `OM_AVG(Income[variant] - Income[base]) , OM_MAX( 100 * (Salary[variant] + Pension[variant]) / Income[base])` and group by two dimension attributes: `AgeGroup , Sex`.

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.

It is also possible to use parameter(s) in calculation, parameter must be a scalar of float or integer type. For example: `OM_COUNT_IF((Income[variant - Income[base]) > param.High[base])`, where `param.High[base]` is a value of scalar parameter `High` in `[base]` model run.

Enum-based microdata attributes returned as enum codes.

Following aggregation functions avaliable:

- `OM_AVG` mean of accumulators sub-values
- `OM_SUM` sum of accumulators sub-values
- `OM_COUNT` count of accumulators sub-values (excluding NULL's)
- `OM_COUNT_IF` count values matching condition
- `OM_MAX` maximum of accumulators sub-values
- `OM_MIN` minimum of accumulators sub-values
- `OM_VAR` variance of accumulators sub-values
- `OM_SD` standard deviation of accumulators sub-values
- `OM_SE` standard error of accumulators sub-values
- `OM_CV` coefficient of variation of accumulators sub-values

For more details please see: Model Output Expressions

Methods:

```
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/compare/:compare/variant/:variant
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/compare/:compare/variant/:variant/start/:start
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/compare/:compare/variant/:variant/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) base model run digest, run stamp or run name
:variant - (required) variant model run(s): comma-separated list of digests, run stamps or run names
```

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
```

:group-by - (required) comma-separated list of dimension attribute(s) to group by aggreagated values, dimension attribute must be enum-based or boolean type.

:compare - (required) comma-separated list of comparions of microdata value attribute(s), value attribute must be float of integer type.

:start - (optional) start "page" row number, zero-based.
:count - (optional) "page" size, number of rows to select, if count <= 0 then all rows selected.

Call examples:

http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/compare/OM_AVG(Income%5Bvariant%5D-Income%5Bbase%5D)/variant/Microdata%20other%20in%20database
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/compare/OM_AVG(Income%5Bvariant%5D-Income%5Bbase%5D)/variant/Microdata%20other%20in%20database/start/2
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/compare/OM_AVG(Income%5Bvariant%5D-Income%5Bbase%5D)/variant/Microdata%20other%20in%20database/start/2/count/4

http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/compare/OM_AVG(Income%5Bvariant%5D-Income%5Bbase%5D),OM_AVG(Salary)/variant/Microdata%20other%20in%20database
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/compare/OM_AVG((Income%5Bvariant%5D-Income%5Bbase%5D)+param.StartingSeed%5Bbase%5D)/variant/Microdata%20other%20in%20database

Note:

- `OM_AVG(Income%5Bvariant%5D-Income%5Bbase%5D)` is URL encoded: `OM_AVG(Income[variant]-Income[base])`.
- `OM_AVG((Income%5Bvariant%5D-Income%5Bbase%5D)+param.StartingSeed%5Bbase%5D)` is URL encoded: `OM_AVG((Income[variant]-Income[base])+param.StartingSeed[base])`.

Return example:

http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/compare/OM_AVG(Income%5Bvariant%5D-Income%5Bbase%5D)/variant/Microdata%20other%20in%20database


```
[
  {
    "Attr": [
      {
        "IsNull": false,
        "Value": "10-20"
      },
      {
        "IsNull": false,
        "Value": "M"
      },
      {
        "IsNull": false,
        "Value": -6701256.201619063
      }
    ],
    "CalcName": "ex_12000",
    "RunDigest": "86135ceed94d1239937a42e088a7fcb7"
  },
  {
    "Attr": [
      {
        "IsNull": false,
        "Value": "10-20"
      },
      {
        "IsNull": false,
        "Value": "F"
      },
      {
        "IsNull": false,
        "Value": -6702689.143904675
      }
    ],
    "CalcName": "ex_12000",
    "RunDigest": "86135ceed94d1239937a42e088a7fcb7"
  },
  {
    "Attr": [
      {
        "IsNull": false,
        "Value": "20-30"
      },
```

```
      {
        "IsNull": false,
        "Value": "M"
      },
      {
        "IsNull": false,
        "Value": -67108864
      }
    ],
    "CalcName": "ex_12000",
    "RunDigest": "86135ceed94d1239937a42e088a7fcb7"
  },
  {
    "Attr": [
      {
        "IsNull": false,
        "Value": "20-30"
      },
      {
        "IsNull": false,
        "Value": "F"
      },
      {
        "IsNull": false,
        "Value": -67112960
      }
    ],
    "CalcName": "ex_12000",
    "RunDigest": "86135ceed94d1239937a42e088a7fcb7"
  },
  {
    "Attr": [
      {
        "IsNull": false,
        "Value": "30-40"
      },
      {
        "IsNull": false,
        "Value": "M"
      },
      {
        "IsNull": false,
        "Value": -67149824
      }
    ],
    "CalcName": "ex_12000",
    "RunDigest": "86135ceed94d1239937a42e088a7fcb7"
  },
  {
    "Attr": [
      {
        "IsNull": false,
        "Value": "30-40"
      },
      {
        "IsNull": false,
        "Value": "F"
      },
      {
        "IsNull": false,
        "Value": -67153920
      }
    ],
    "CalcName": "ex_12000",
    "RunDigest": "86135ceed94d1239937a42e088a7fcb7"
  },
  {
    "Attr": [
      {
        "IsNull": false,
        "Value": "40+"
      },
      {
        "IsNull": false,
        "Value": "M"
      },
      {
        "IsNull": false,
        "Value": -37327458.47071971
      }
    ],
    "CalcName": "ex_12000",
    "RunDigest": "86135ceed94d1239937a42e088a7fcb7"
  },
  {
    "Attr": [
      {
        "IsNull": false
```

```
      "IsNull": false,
      "Value": "40+"
    },
    {
      "IsNull": false,
      "Value": "F"
    },
    {
      "IsNull": false,
      "Value": -35538991.09092548
    }
  ],
  "CalcName": "ex_12000",
  "RunDigest": "86135ceed94d1239937a42e088a7fcb7"
}
]
```

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 calculated table expressions from model run

Calculate and read output table expression(s) values from model run as csv file.

Read output table expressions, calculate additional measure for each expression and get it as response stream UTF-8 outputTable.csv file attachment, optionally starts with byte order mark (BOM).

Measures calculated as one of the following:

- for each table expression calculate one of: avg, sum, count, max, min, var, sd, se, cv
- as arbitrary aggreagated expressions provided as comma separated list

Dimension(s) returned as enum codes.

Methods:

```
GET /api/model/:model/run/:run/table/:name/calc/:calc/csv
GET /api/model/:model/run/:run/table/:name/calc/:calc/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
```

```
:calc - (required) name of additional measure to calculate
```

Additional measure must be one of:

- `avg` mean of expression sub-values
- `sum` sum of expression sub-values
- `count` count of expression sub-values (excluding NULL's)
- `max` maximum of expression sub-values
- `min` minimum of expression sub-values
- `var` variance of expression sub-values
- `sd` standard deviation of expression sub-values
- `se` standard error of expression sub-values
- `cv` coefficient of variation of expression sub-values

Or a list of comma-separated aggreagated expressions, for example: OM_AVG(acc0) , 2 * SQRT(OM_SUM(acc1) - OM_SD(acc0))

It is also possible to use parameter(s) in calculation, parameter must be a scalar of float or integer type. For example: `OM_COUNT_IF(acc1 > param.High)` , where `param.High` is a value of scalar parameter `High` in that model run.

Following aggregation functions avaliable:

- `OM_AVG` mean of accumulators sub-values
- `OM_SUM` sum of accumulators sub-values
- `OM_COUNT` count of accumulators sub-values (excluding NULL's)
- `OM_COUNT_IF` count values matching condition
- `OM_MAX` maximum of accumulators sub-values
- `OM_MIN` minimum of accumulators sub-values
- `OM_VAR` variance of accumulators sub-values
- `OM_SD` standard deviation of accumulators sub-values
- `OM_SE` standard error of accumulators sub-values
- `OM_CV` coefficient of variation of accumulators sub-values

For more details please see: Model Output Expressions

Call examples:

```
http://localhost:4040/api/model/RiskPaths/run/RiskPaths_Default/table/T04_FertilityRatesByAgeGroup/calc/avg/csv
http://localhost:4040/api/model/RiskPaths/run/RiskPaths_Default/table/T04_FertilityRatesByAgeGroup/calc/sd/csv-bom
http://localhost:4040/api/model/_201208171604590148_/run/f172e98da17beb058f30f11768053456/table/salarySex/calc/se/csv
http://localhost:4040/api/model/_201208171604590148_/run/2019_01_17_19_59_52_998/table/salarySex/calc/cv/csv

http://localhost:4040/api/model/modelOne/run/Default-4/table/salarySex/calc/OM_AVG(acc0),2*SQRT(OM_SUM(acc1)-OM_SD(acc0))/csv
http://localhost:4040/api/model/modelOne/run/Default-4/table/salarySex/calc/OM_COUNT_IF(acc0%3Cparam.StartingSeed)/csv
```

Note: `OM_COUNT_IF(acc0%3Cparam.StartingSeed)` is URL encoded: `OM_COUNT_IF(acc0<param.StartingSeed)`

Return example:

```
curl http://localhost:4040/api/model/RiskPaths/run/RiskPaths_Default_4/table/T04_FertilityRatesByAgeGroup/calc/avg/csv
```

```
run_digest,calc_name,Dim0,Dim1,calc_value
b794d3399099035740e117378c523feb,Expr0,"(-∞,15)",US_NEVER_IN_UNION,0
b794d3399099035740e117378c523feb,Expr0,"(-∞,15)",US_FIRST_UNION_PERIOD1,null
b794d3399099035740e117378c523feb,Expr0,"(-∞,15)",US_FIRST_UNION_PERIOD2,null
b794d3399099035740e117378c523feb,Expr0,"(-∞,15)",US_AFTER_FIRST_UNION,null
b794d3399099035740e117378c523feb,Expr0,"(-∞,15)",US_SECOND_UNION,null
b794d3399099035740e117378c523feb,Expr0,"(-∞,15)",US_AFTER_SECOND_UNION,null
b794d3399099035740e117378c523feb,Expr0,"[15,17.5)",US_NEVER_IN_UNION,0.0183139978773642
b794d3399099035740e117378c523feb,Expr0,"[15,17.5)",US_FIRST_UNION_PERIOD1,0.315246747433367
b794d3399099035740e117378c523feb,Expr0,"[15,17.5)",US_FIRST_UNION_PERIOD2,null
b794d3399099035740e117378c523feb,Expr0,"[15,17.5)",US_AFTER_FIRST_UNION,0
b794d3399099035740e117378c523feb,Expr0,"[15,17.5)",US_SECOND_UNION,0
b794d3399099035740e117378c523feb,Expr0,"[15,17.5)",US_AFTER_SECOND_UNION,null
b794d3399099035740e117378c523feb,Expr0,"[17.5,20)",US_NEVER_IN_UNION,0.0537541294539804
b794d3399099035740e117378c523feb,Expr0,"[17.5,20)",US_FIRST_UNION_PERIOD1,0.713129647941936
b794d3399099035740e117378c523feb,Expr0,"[17.5,20)",US_FIRST_UNION_PERIOD2,0.27913884088947
b794d3399099035740e117378c523feb,Expr0,"[17.5,20)",US_AFTER_FIRST_UNION,0.0340306112997705
b794d3399099035740e117378c523feb,Expr0,"[17.5,20)",US_SECOND_UNION,0.631356771396228
b794d3399099035740e117378c523feb,Expr0,"[17.5,20)",US_AFTER_SECOND_UNION,null
b794d3399099035740e117378c523feb,Expr0,"[20,22.5)",US_NEVER_IN_UNION,0.0544540553970037
b794d3399099035740e117378c523feb,Expr0,"[20,22.5)",US_FIRST_UNION_PERIOD1,0.814226115392992
b794d3399099035740e117378c523feb,Expr0,"[20,22.5)",US_FIRST_UNION_PERIOD2,0.225999767108206
b794d3399099035740e117378c523feb,Expr0,"[20,22.5)",US_AFTER_FIRST_UNION,0.0225289422479095
b794d3399099035740e117378c523feb,Expr0,"[20,22.5)",US_SECOND_UNION,0.552801004995511
b794d3399099035740e117378c523feb,Expr0,"[20,22.5)",US_AFTER_SECOND_UNION,0
b794d3399099035740e117378c523feb,Expr0,"[22.5,25)",US_NEVER_IN_UNION,0.0539099306643693
b794d3399099035740e117378c523feb,Expr0,"[22.5,25)",US_FIRST_UNION_PERIOD1,0.809582230266138
b794d3399099035740e117378c523feb,Expr0,"[22.5,25)",US_FIRST_UNION_PERIOD2,0.195155708061047
b794d3399099035740e117378c523feb,Expr0,"[22.5,25)",US_AFTER_FIRST_UNION,0.0414901238939861
b794d3399099035740e117378c523feb,Expr0,"[22.5,25)",US_SECOND_UNION,0.59945794749404
b794d3399099035740e117378c523feb,Expr0,"[22.5,25)",US_AFTER_SECOND_UNION,0
b794d3399099035740e117378c523feb,Expr0,"[25,27.5)",US_NEVER_IN_UNION,0.0415985067709889
b794d3399099035740e117378c523feb,Expr0,"[25,27.5)",US_FIRST_UNION_PERIOD1,0.602459144412015
b794d3399099035740e117378c523feb,Expr0,"[25,27.5)",US_FIRST_UNION_PERIOD2,0.187080449150788
b794d3399099035740e117378c523feb,Expr0,"[25,27.5)",US_AFTER_FIRST_UNION,0.0203496069483281
b794d3399099035740e117378c523feb,Expr0,"[25,27.5)",US_SECOND_UNION,1.00906724659711
b794d3399099035740e117378c523feb,Expr0,"[25,27.5)",US_AFTER_SECOND_UNION,0
b794d3399099035740e117378c523feb,Expr0,"[27.5,30)",US_NEVER_IN_UNION,0.0314673689946165
b794d3399099035740e117378c523feb,Expr0,"[27.5,30)",US_FIRST_UNION_PERIOD1,0.462198924442823
```

b794d3399099035740e117378c523feb,Expr0,"[27.5,30)",US_FIRST_UNION_PERIOD2,0.137851629822403
b794d3399099035740e117378c523feb,Expr0,"[27.5,30)",US_AFTER_FIRST_UNION,0.0229824909841167
b794d3399099035740e117378c523feb,Expr0,"[27.5,30)",US_SECOND_UNION,0.362222813716797
b794d3399099035740e117378c523feb,Expr0,"[27.5,30)",US_AFTER_SECOND_UNION,0.0572026776260319
b794d3399099035740e117378c523feb,Expr0,"[30,32.5)",US_NEVER_IN_UNION,0.0416497396600501
b794d3399099035740e117378c523feb,Expr0,"[30,32.5)",US_FIRST_UNION_PERIOD1,0.549536875782365
b794d3399099035740e117378c523feb,Expr0,"[30,32.5)",US_FIRST_UNION_PERIOD2,0.110088801281195
b794d3399099035740e117378c523feb,Expr0,"[30,32.5)",US_AFTER_FIRST_UNION,0.0468326974687102
b794d3399099035740e117378c523feb,Expr0,"[30,32.5)",US_SECOND_UNION,0.330292197388013
b794d3399099035740e117378c523feb,Expr0,"[30,32.5)",US_AFTER_SECOND_UNION,0
b794d3399099035740e117378c523feb,Expr0,"[32.5,35)",US_NEVER_IN_UNION,0.0221476827597504
b794d3399099035740e117378c523feb,Expr0,"[32.5,35)",US_FIRST_UNION_PERIOD1,0.240119203514267
b794d3399099035740e117378c523feb,Expr0,"[32.5,35)",US_FIRST_UNION_PERIOD2,0.0732205051114689
b794d3399099035740e117378c523feb,Expr0,"[32.5,35)",US_AFTER_FIRST_UNION,0
b794d3399099035740e117378c523feb,Expr0,"[32.5,35)",US_SECOND_UNION,0.391609286784764
b794d3399099035740e117378c523feb,Expr0,"[32.5,35)",US_AFTER_SECOND_UNION,0
b794d3399099035740e117378c523feb,Expr0,"[35,37.5)",US_NEVER_IN_UNION,0.0118488890311401
b794d3399099035740e117378c523feb,Expr0,"[35,37.5)",US_FIRST_UNION_PERIOD1,0.131862730376457
b794d3399099035740e117378c523feb,Expr0,"[35,37.5)",US_FIRST_UNION_PERIOD2,0.0830998727570066
b794d3399099035740e117378c523feb,Expr0,"[35,37.5)",US_AFTER_FIRST_UNION,0.0221047417383149
b794d3399099035740e117378c523feb,Expr0,"[35,37.5)",US_SECOND_UNION,0.279863987999404
b794d3399099035740e117378c523feb,Expr0,"[35,37.5)",US_AFTER_SECOND_UNION,0.114442750608107
b794d3399099035740e117378c523feb,Expr0,"[37.5,40)",US_NEVER_IN_UNION,0.017525081049452
b794d3399099035740e117378c523feb,Expr0,"[37.5,40)",US_FIRST_UNION_PERIOD1,0.157747779554811
b794d3399099035740e117378c523feb,Expr0,"[37.5,40)",US_FIRST_UNION_PERIOD2,0.0572711925357155
b794d3399099035740e117378c523feb,Expr0,"[37.5,40)",US_AFTER_FIRST_UNION,0
b794d3399099035740e117378c523feb,Expr0,"[37.5,40)",US_SECOND_UNION,0.169455199059931
b794d3399099035740e117378c523feb,Expr0,"[37.5,40)",US_AFTER_SECOND_UNION,0
b794d3399099035740e117378c523feb,Expr0,"[40,∞)",US_NEVER_IN_UNION,0
b794d3399099035740e117378c523feb,Expr0,"[40,∞)",US_FIRST_UNION_PERIOD1,0
b794d3399099035740e117378c523feb,Expr0,"[40,∞)",US_FIRST_UNION_PERIOD2,0
b794d3399099035740e117378c523feb,Expr0,"[40,∞)",US_AFTER_FIRST_UNION,0
b794d3399099035740e117378c523feb,Expr0,"[40,∞)",US_SECOND_UNION,0
b794d3399099035740e117378c523feb,Expr0,"[40,∞)",US_AFTER_SECOND_UNION,0
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"(-∞,15)",US_NEVER_IN_UNION,0
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"(-∞,15)",US_FIRST_UNION_PERIOD1,null
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"(-∞,15)",US_FIRST_UNION_PERIOD2,null
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"(-∞,15)",US_AFTER_FIRST_UNION,null
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"(-∞,15)",US_SECOND_UNION,null
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"(-∞,15)",US_AFTER_SECOND_UNION,null
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[15,17.5)",US_NEVER_IN_UNION,0.0183150153186944
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[15,17.5)",US_FIRST_UNION_PERIOD1,0.316271343152912
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[15,17.5)",US_FIRST_UNION_PERIOD2,null
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[15,17.5)",US_AFTER_FIRST_UNION,0
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[15,17.5)",US_SECOND_UNION,0
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[15,17.5)",US_AFTER_SECOND_UNION,null
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[17.5,20)",US_NEVER_IN_UNION,0.0537675746731247
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[17.5,20)",US_FIRST_UNION_PERIOD1,0.713529542211133
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[17.5,20)",US_FIRST_UNION_PERIOD2,0.290605057165821
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[17.5,20)",US_AFTER_FIRST_UNION,0.0257953477423063
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[17.5,20)",US_SECOND_UNION,0.971013422710935
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[17.5,20)",US_AFTER_SECOND_UNION,null
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[20,22.5)",US_NEVER_IN_UNION,0.054451507853128
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[20,22.5)",US_FIRST_UNION_PERIOD1,0.815769510398502
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[20,22.5)",US_FIRST_UNION_PERIOD2,0.229287238004696
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[20,22.5)",US_AFTER_FIRST_UNION,0.0127116167313564
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[20,22.5)",US_SECOND_UNION,0.429415814994704
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[20,22.5)",US_AFTER_SECOND_UNION,0
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[22.5,25)",US_NEVER_IN_UNION,0.0541040962861145
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[22.5,25)",US_FIRST_UNION_PERIOD1,0.80980843837352
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[22.5,25)",US_FIRST_UNION_PERIOD2,0.195232130104512
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[22.5,25)",US_AFTER_FIRST_UNION,0.0425462458785113
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[22.5,25)",US_SECOND_UNION,0.831967409785808
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[22.5,25)",US_AFTER_SECOND_UNION,0
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[25,27.5)",US_NEVER_IN_UNION,0.0417067099443355
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[25,27.5)",US_FIRST_UNION_PERIOD1,0.602467976857289
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[25,27.5)",US_FIRST_UNION_PERIOD2,0.186968206655534
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[25,27.5)",US_AFTER_FIRST_UNION,0.0315133813846249
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[25,27.5)",US_SECOND_UNION,1.28700220240812
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[25,27.5)",US_AFTER_SECOND_UNION,0
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[27.5,30)",US_NEVER_IN_UNION,0.0314820070793188
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[27.5,30)",US_FIRST_UNION_PERIOD1,0.462127063133201
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[27.5,30)",US_FIRST_UNION_PERIOD2,0.137897098019438
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[27.5,30)",US_AFTER_FIRST_UNION,0.0295557293946093
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[27.5,30)",US_SECOND_UNION,0.268318808255325
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[27.5,30)",US_AFTER_SECOND_UNION,0.0267057657548253
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[30,32.5)",US_NEVER_IN_UNION,0.0416625733706113
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[30,32.5)",US_FIRST_UNION_PERIOD1,0.567544621403048
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[30,32.5)",US_FIRST_UNION_PERIOD2,0.111089012753633
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[30,32.5)",US_AFTER_FIRST_UNION,0.0452064833182238
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[30,32.5)",US_SECOND_UNION,0.463879691400302
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[30,32.5)",US_AFTER_SECOND_UNION,0
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[32.5,35)",US_NEVER_IN_UNION,0.022442211806591
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[32.5,35)",US_FIRST_UNION_PERIOD1,0.26518578732689
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[32.5,35)",US_FIRST_UNION_PERIOD2,0.0759259959848827
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[32.5,35)",US_AFTER_FIRST_UNION,0
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[32.5,35)",US_SECOND_UNION,0.557998028374596
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[32.5,35)",US_AFTER_SECOND_UNION,0

b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[35,37.5)",US_NEVER_IN_UNION,0.0110885846008521
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[35,37.5)",US_FIRST_UNION_PERIOD1,0.13577342261578
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[35,37.5)",US_FIRST_UNION_PERIOD2,0.0815471990733788
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[35,37.5)",US_AFTER_FIRST_UNION,0.0123398583333944
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[35,37.5)",US_SECOND_UNION,0.235867044941869
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[35,37.5)",US_AFTER_SECOND_UNION,0.0534359868184371
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[37.5,40)",US_NEVER_IN_UNION,0.0172608633104785
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[37.5,40)",US_FIRST_UNION_PERIOD1,0.165574819914602
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[37.5,40)",US_FIRST_UNION_PERIOD2,0.0569355432493007
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[37.5,40)",US_AFTER_FIRST_UNION,0
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[37.5,40)",US_SECOND_UNION,0.165135000822642
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[37.5,40)",US_AFTER_SECOND_UNION,0
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[40,∞)",US_NEVER_IN_UNION,0
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[40,∞)",US_FIRST_UNION_PERIOD1,0
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[40,∞)",US_FIRST_UNION_PERIOD2,0
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[40,∞)",US_AFTER_FIRST_UNION,0
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[40,∞)",US_SECOND_UNION,0
b794d3399099035740e117378c523feb,OM_AVG_Expr0,"[40,∞)",US_AFTER_SECOND_UNION,0

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

Calculate and read output table expression(s) values from model run as csv file.

Read output table expressions, calculate additional measure for each expression and get it as response stream UTF-8 outputTable.csv file attachment, optionally starts with byte order mark (BOM).

Measures calculated as one of the following:

- for each table expression calculate one of: avg, sum, count, max, min, var, sd, se, cv
- as arbitrary aggreagated expressions provided as comma separated list

Dimension(s) returned as enum id's.

Methods:

```
GET /api/model/:model/run/:run/table/:name/calc/:calc/csv-id
GET /api/model/:model/run/:run/table/:name/calc/:calc/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
```

```
:calc - (required) name of additional measure to calculate
```

Additional measure must be one of:

- `avg` mean of expression sub-values
- `sum` sum of expression sub-values
- `count` count of expression sub-values (excluding NULL's)
- `max` maximum of expression sub-values
- `min` minimum of expression sub-values
- `var` variance of expression sub-values
- `sd` standard deviation of expression sub-values
- `se` standard error of expression sub-values
- `cv` coefficient of variation of expression sub-values

Or a list of comma-separated aggreagated expressions, for example: OM_AVG(acc0) , 2 * SQRT(OM_SUM(acc1) - OM_SD(acc0))

It is also possible to use parameter(s) in calculation, parameter must be a scalar of float or integer type. For example: `OM_COUNT_IF(acc1 > param.High)` , where `param.High` is a value of scalar parameter `High` in that model run.

Following aggregation functions avaliable:

- `OM_AVG` mean of accumulators sub-values
- `OM_SUM` sum of accumulators sub-values
- `OM_COUNT` count of accumulators sub-values (excluding NULL's)
- `OM_COUNT_IF` count values matching condition
- `OM_MAX` maximum of accumulators sub-values
- `OM_MIN` minimum of accumulators sub-values
- `OM_VAR` variance of accumulators sub-values
- `OM_SD` standard deviation of accumulators sub-values
- `OM_SE` standard error of accumulators sub-values
- `OM_CV` coefficient of variation of accumulators sub-values

For more details please see: Model Output Expressions

Call examples:

```
http://localhost:4040/api/model/RiskPaths/run/RiskPaths_Default/table/T04_FertilityRatesByAgeGroup/calc/avg/csv-id
http://localhost:4040/api/model/RiskPaths/run/RiskPaths_Default/table/T04_FertilityRatesByAgeGroup/calc/sd/csv-id-bom
http://localhost:4040/api/model/_201208171604590148_/run/f172e98da17beb058f30f11768053456/table/salarySex/calc/se/csv-id
http://localhost:4040/api/model/_201208171604590148_/run/2019_01_17_19_59_52_998/table/salarySex/calc/cv/csv-id-bom

http://localhost:4040/api/model/modelOne/run/Default-4/table/salarySex/calc/OM_AVG(acc0),2*SQRT(OM_SUM(acc1)-OM_SD(acc0))/csv-id
http://localhost:4040/api/model/modelOne/run/Default-4/table/salarySex/calc/OM_COUNT_IF(acc0%3Cparam.StartingSeed)/csv-id
```

Note: `OM_COUNT_IF(acc0%3Cparam.StartingSeed)` is URL encoded: `OM_COUNT_IF(acc0<param.StartingSeed)`

Return example:

`calc_id` column contains output table expression id's: `0 <= expr_id < 12000` or id's of calculated values: `expr_id + 12000`. For example, id of calculated value `SE(Expr2)` is: `calc_id = 12002`

```
curl http://localhost:4040/api/model/RiskPaths/run/RiskPaths_Default_4/table/T04_FertilityRatesByAgeGroup/calc/avg/csv-id
```

```
run_id,calc_id,Dim0,Dim1,calc_value
102,0,0,0,0
102,0,0,1,null
102,0,0,2,null
102,0,0,3,null
102,0,0,4,null
102,0,0,5,null
102,0,1,0,0.0183139978773642
102,0,1,1,0.315246747433367
102,0,1,2,null
102,0,1,3,0
102,0,1,4,0
102,0,1,5,null
102,0,2,0,0.0537541294539804
102,0,2,1,0.713129647941936
102,0,2,2,0.27913884088947
102,0,2,3,0.0340306112997705
102,0,2,4,0.631356771396228
102,0,2,5,null
102,0,3,0,0.0544540553970037
102,0,3,1,0.814226115392992
102,0,3,2,0.225999767108206
102,0,3,3,0.0225289422479095
102,0,3,4,0.552801004995511
102,0,3,5,0
102,0,4,0,0.0539099306643693
102,0,4,1,0.809582230266138
102,0,4,2,0.195155708061047
102,0,4,3,0.0414901238939861
102,0,4,4,0.59945794749404
102,0,4,5,0
102,0,5,0,0.0415985067709889
102,0,5,1,0.602459144412015
102,0,5,2,0.187080449150788
102,0,5,3,0.0203496069483281
```

102,0,5,4,1.00906724659711
102,0,5,5,0
102,0,6,0,0.0314673689946165
102,0,6,1,0.462198924442823
102,0,6,2,0.137851629822403
102,0,6,3,0.0229824909841167
102,0,6,4,0.362222813716797
102,0,6,5,0.0572026776260319
102,0,7,0,0.0416497396600501
102,0,7,1,0.549536875782365
102,0,7,2,0.110088801281195
102,0,7,3,0.0468326974687102
102,0,7,4,0.330292197388013
102,0,7,5,0
102,0,8,0,0.0221476827597504
102,0,8,1,0.240119203514267
102,0,8,2,0.0732205051114689
102,0,8,3,0
102,0,8,4,0.391609286784764
102,0,8,5,0
102,0,9,0,0.0118488890311401
102,0,9,1,0.131862730376457
102,0,9,2,0.0830998727570066
102,0,9,3,0.0221047417383149
102,0,9,4,0.279863987999404
102,0,9,5,0.114442750608107
102,0,10,0,0.017525081049452
102,0,10,1,0.157747779554811
102,0,10,2,0.0572711925357155
102,0,10,3,0
102,0,10,4,0.169455199059931
102,0,10,5,0
102,0,11,0,0
102,0,11,1,0
102,0,11,2,0
102,0,11,3,0
102,0,11,4,0
102,0,11,5,0
102,12000,0,0,0
102,12000,0,1,null
102,12000,0,2,null
102,12000,0,3,null
102,12000,0,4,null
102,12000,0,5,null
102,12000,1,0,0.0183150153186944
102,12000,1,1,0.316271343152912
102,12000,1,2,null
102,12000,1,3,0
102,12000,1,4,0
102,12000,1,5,null
102,12000,2,0,0.0537675746731247
102,12000,2,1,0.713529542211133
102,12000,2,2,0.290605057165821
102,12000,2,3,0.0257953477423063
102,12000,2,4,0.971013422710935
102,12000,2,5,null
102,12000,3,0,0.054451507853128
102,12000,3,1,0.815769510398502
102,12000,3,2,0.229287238004696
102,12000,3,3,0.0127116167313564
102,12000,3,4,0.429415814994704
102,12000,3,5,0
102,12000,4,0,0.0541040962861145
102,12000,4,1,0.80980843837352
102,12000,4,2,0.195232130104512
102,12000,4,3,0.0425462458785113
102,12000,4,4,0.831967409785808
102,12000,4,5,0
102,12000,5,0,0.0417067099443355
102,12000,5,1,0.602467976857289
102,12000,5,2,0.186968206655534
102,12000,5,3,0.0315133813846249
102,12000,5,4,1.28700220240812
102,12000,5,5,0
102,12000,6,0,0.0314820070793188
102,12000,6,1,0.462127063133201
102,12000,6,2,0.137897098019438
102,12000,6,3,0.0295557293946093
102,12000,6,4,0.268318808255325
102,12000,6,5,0.0267057657548253
102,12000,7,0,0.0416625733706113
102,12000,7,1,0.567544621403048
102,12000,7,2,0.111089012753633
102,12000,7,3,0.0452064833182238
102,12000,7,4,0.463879691400302
102,12000,7,5,0
102,12000,8,0,0.022442211806591
102,12000,8,1,0.26518578732689

```
102,12000,8,2,0.0759259959848827
102,12000,8,3,0
102,12000,8,4,0.557998028374596
102,12000,8,5,0
102,12000,9,0,0.0110885846008521
102,12000,9,1,0.13577342261578
102,12000,9,2,0.0815471990733788
102,12000,9,3,0.0123398583333944
102,12000,9,4,0.235867044941869
102,12000,9,5,0.0534359868184371
102,12000,10,0,0.0172608633104785
102,12000,10,1,0.165574819914602
102,12000,10,2,0.0569355432493007
102,12000,10,3,0
102,12000,10,4,0.165135000822642
102,12000,10,5,0
102,12000,11,0,0
102,12000,11,1,0
102,12000,11,2,0
102,12000,11,3,0
102,12000,11,4,0
102,12000,11,5,0
```

GET csv model runs comparison table expressions

Compare model runs and return results as csv file.

Compare `[base]` and `[variant]` model runs output values for each expression and get it as response stream UTF-8 outputTable.csv file attachment, optionally starts with byte order mark (BOM).

Comparison can be calculated as one of the following:

- for each table expression use one of: `diff`, `ratio` or `percent` comparison between `[base]` and `[variant]` model runs.
- use comma separated list of comparison expressions between `[base]` and `[variant]` or simple expression for each run.

Dimension(s) returned as enum codes.

Methods:

```
GET /api/model/:model/run/:run/table/:name/compare/:compare/variant/:variant/csv
GET /api/model/:model/run/:run/table/:name/compare/:compare/variant/:variant/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) base model run digest, run stamp or run name
:variant - (required) variant model run(s): comma-separated list of digests, run stamps or run names
```

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
```

```
:compare - (required) comparison to calculate
```

- `diff` differenece of values between variant and base run, e.g.: `Expr0[variant] - Expr0[base]`
- `ratio` ratio of values between variant and base run, e.g.: `Expr0[variant] / Expr0[base]`
- `percent` proportional difference multipled by 100, e.g.: `100 * (Expr0[variant] - Expr0[base]) / Expr0[base]`

Or a list of comma-separated expressions, for example: `expr0, expr1[variant] + expr2[base]`

It is also possible to use parameter(s) in calculation, parameter must be a scalar of float or integer type. For example: `(Expr1[variant] - Expr1[base]) * param.Scale[base]`, where `param.Scale` is a value of scalar parameter `Scale` in `[base]` model run.

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/compare/diff/variant/Default-4/csv
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/compare/ratio/variant/Default-4/csv
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/compare/percent/variant/Default-4/csv

http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/compare/diff/variant/Default-4/csv-bom

http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/compare/expr0,expr1,expr2/variant/Default-4/csv
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/compare/expr0%2Cexpr1%5Bvariant%5D%2Bexpr2%5Bbase%5D/variant/Default-4/csv
```

Note: above `expr0%2Cexpr1%5Bvariant%5D%2Bexpr2%5Bbase%5D` is URL encoded: `expr0,expr1[variant]+expr2[base]`.

Example:

```
curl http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/compare/ratio/variant/Default-4,Sub-values_4/csv
```

```
run_digest,calc_name,dim0,dim1,calc_value
ca663651953bae94d0afdf71edba4c91,expr0,L,M,50
ca663651953bae94d0afdf71edba4c91,expr0,L,F,60
ca663651953bae94d0afdf71edba4c91,expr0,L,all,1
ca663651953bae94d0afdf71edba4c91,expr0,M,M,51.6
ca663651953bae94d0afdf71edba4c91,expr0,M,F,62
ca663651953bae94d0afdf71edba4c91,expr0,M,all,2
ca663651953bae94d0afdf71edba4c91,expr0,H,M,53.2
ca663651953bae94d0afdf71edba4c91,expr0,H,F,64
ca663651953bae94d0afdf71edba4c91,expr0,H,all,3
ca663651953bae94d0afdf71edba4c91,expr1,L,M,0
ca663651953bae94d0afdf71edba4c91,expr1,L,F,1
ca663651953bae94d0afdf71edba4c91,expr1,L,all,800
ca663651953bae94d0afdf71edba4c91,expr1,M,M,1
ca663651953bae94d0afdf71edba4c91,expr1,M,F,2
ca663651953bae94d0afdf71edba4c91,expr1,M,all,801
ca663651953bae94d0afdf71edba4c91,expr1,H,M,2
ca663651953bae94d0afdf71edba4c91,expr1,H,F,3
ca663651953bae94d0afdf71edba4c91,expr1,H,all,802
ca663651953bae94d0afdf71edba4c91,expr2,L,M,50
ca663651953bae94d0afdf71edba4c91,expr2,L,F,61
ca663651953bae94d0afdf71edba4c91,expr2,L,all,801
ca663651953bae94d0afdf71edba4c91,expr2,M,M,52.6
ca663651953bae94d0afdf71edba4c91,expr2,M,F,64
ca663651953bae94d0afdf71edba4c91,expr2,M,all,803
ca663651953bae94d0afdf71edba4c91,expr2,H,M,55.2
ca663651953bae94d0afdf71edba4c91,expr2,H,F,67
ca663651953bae94d0afdf71edba4c91,expr2,H,all,805
ca663651953bae94d0afdf71edba4c91,expr3,L,M,0
ca663651953bae94d0afdf71edba4c91,expr3,L,F,60
ca663651953bae94d0afdf71edba4c91,expr3,L,all,800
ca663651953bae94d0afdf71edba4c91,expr3,M,M,51.6
ca663651953bae94d0afdf71edba4c91,expr3,M,F,124
ca663651953bae94d0afdf71edba4c91,expr3,M,all,1602
ca663651953bae94d0afdf71edba4c91,expr3,H,M,106.4
ca663651953bae94d0afdf71edba4c91,expr3,H,F,192
ca663651953bae94d0afdf71edba4c91,expr3,H,all,2406
c519fc5869f244ac4c80ae44695a4272,expr0,L,M,50
c519fc5869f244ac4c80ae44695a4272,expr0,L,F,60
c519fc5869f244ac4c80ae44695a4272,expr0,L,all,1201
c519fc5869f244ac4c80ae44695a4272,expr0,M,M,51.6
c519fc5869f244ac4c80ae44695a4272,expr0,M,F,62
c519fc5869f244ac4c80ae44695a4272,expr0,M,all,1202
c519fc5869f244ac4c80ae44695a4272,expr0,H,M,53.2
c519fc5869f244ac4c80ae44695a4272,expr0,H,F,64
c519fc5869f244ac4c80ae44695a4272,expr0,H,all,1203
c519fc5869f244ac4c80ae44695a4272,expr1,L,M,6
c519fc5869f244ac4c80ae44695a4272,expr1,L,F,10
c519fc5869f244ac4c80ae44695a4272,expr1,L,all,3206
c519fc5869f244ac4c80ae44695a4272,expr1,M,M,10
c519fc5869f244ac4c80ae44695a4272,expr1,M,F,14
c519fc5869f244ac4c80ae44695a4272,expr1,M,all,3210
c519fc5869f244ac4c80ae44695a4272,expr1,H,M,14
c519fc5869f244ac4c80ae44695a4272,expr1,H,F,18
c519fc5869f244ac4c80ae44695a4272,expr1,H,all,3214
c519fc5869f244ac4c80ae44695a4272,expr2,L,M,51.5
c519fc5869f244ac4c80ae44695a4272,expr2,L,F,62.5
c519fc5869f244ac4c80ae44695a4272,expr2,L,all,2002.5
c519fc5869f244ac4c80ae44695a4272,expr2,M,M,54.1
c519fc5869f244ac4c80ae44695a4272,expr2,M,F,65.5
c519fc5869f244ac4c80ae44695a4272,expr2,M,all,2004.5
c519fc5869f244ac4c80ae44695a4272,expr2,H,M,56.7
c519fc5869f244ac4c80ae44695a4272,expr2,H,F,68.5
c519fc5869f244ac4c80ae44695a4272,expr2,H,all,2006.5
c519fc5869f244ac4c80ae44695a4272,expr3,L,M,75
c519fc5869f244ac4c80ae44695a4272,expr3,L,F,150
c519fc5869f244ac4c80ae44695a4272,expr3,L,all,963601.5
c519fc5869f244ac4c80ae44695a4272,expr3,M,M,129
c519fc5869f244ac4c80ae44695a4272,expr3,M,F,217
c519fc5869f244ac4c80ae44695a4272,expr3,M,all,965605
c519fc5869f244ac4c80ae44695a4272,expr3,H,M,186.2
c519fc5869f244ac4c80ae44695a4272,expr3,H,F,288
c519fc5869f244ac4c80ae44695a4272,expr3,H,all,967610.5
c519fc5869f244ac4c80ae44695a4272,ratio_expr0,L,M,1
c519fc5869f244ac4c80ae44695a4272,ratio_expr0,L,F,1
c519fc5869f244ac4c80ae44695a4272,ratio_expr0,L,all,1201
c519fc5869f244ac4c80ae44695a4272,ratio_expr0,M,M,1
c519fc5869f244ac4c80ae44695a4272,ratio_expr0,M,F,1
c519fc5869f244ac4c80ae44695a4272,ratio_expr0,M,all,601
c519fc5869f244ac4c80ae44695a4272,ratio_expr0,H,M,1
c519fc5869f244ac4c80ae44695a4272,ratio_expr0,H,F,1
c519fc5869f244ac4c80ae44695a4272,ratio_expr0,H,all,401
c519fc5869f244ac4c80ae44695a4272,ratio_expr1,L,M,null
```

```
c519fc5869f244ac4c80ae44695a4272,ratio_expr1,L,F,10
c519fc5869f244ac4c80ae44695a4272,ratio_expr1,L,all,4.0075
c519fc5869f244ac4c80ae44695a4272,ratio_expr1,M,M,10
c519fc5869f244ac4c80ae44695a4272,ratio_expr1,M,F,7
c519fc5869f244ac4c80ae44695a4272,ratio_expr1,M,all,4.00749063670412
c519fc5869f244ac4c80ae44695a4272,ratio_expr1,H,M,7
c519fc5869f244ac4c80ae44695a4272,ratio_expr1,H,F,6
c519fc5869f244ac4c80ae44695a4272,ratio_expr1,H,all,4.0074812967581
c519fc5869f244ac4c80ae44695a4272,ratio_expr2,L,M,1.03
c519fc5869f244ac4c80ae44695a4272,ratio_expr2,L,F,1.02459016393443
c519fc5869f244ac4c80ae44695a4272,ratio_expr2,L,all,2.5
c519fc5869f244ac4c80ae44695a4272,ratio_expr2,M,M,1.02851711026616
c519fc5869f244ac4c80ae44695a4272,ratio_expr2,M,F,1.0234375
c519fc5869f244ac4c80ae44695a4272,ratio_expr2,M,all,2.49626400996264
c519fc5869f244ac4c80ae44695a4272,ratio_expr2,H,M,1.02717391304348
c519fc5869f244ac4c80ae44695a4272,ratio_expr2,H,F,1.02238805970149
c519fc5869f244ac4c80ae44695a4272,ratio_expr2,H,all,2.49254658385093
c519fc5869f244ac4c80ae44695a4272,ratio_expr3,L,M,null
c519fc5869f244ac4c80ae44695a4272,ratio_expr3,L,F,2.5
c519fc5869f244ac4c80ae44695a4272,ratio_expr3,L,all,1204.501875
c519fc5869f244ac4c80ae44695a4272,ratio_expr3,M,M,2.5
c519fc5869f244ac4c80ae44695a4272,ratio_expr3,M,F,1.75
c519fc5869f244ac4c80ae44695a4272,ratio_expr3,M,all,602.749687890137
c519fc5869f244ac4c80ae44695a4272,ratio_expr3,H,M,1.75
c519fc5869f244ac4c80ae44695a4272,ratio_expr3,H,F,1.5
c519fc5869f244ac4c80ae44695a4272,ratio_expr3,H,all,402.165627597672
ffad6e8ed4449dafe11b82c7aea82f74,expr0,L,M,50
ffad6e8ed4449dafe11b82c7aea82f74,expr0,L,F,60
ffad6e8ed4449dafe11b82c7aea82f74,expr0,L,all,1201
ffad6e8ed4449dafe11b82c7aea82f74,expr0,M,M,51.6
ffad6e8ed4449dafe11b82c7aea82f74,expr0,M,F,62
ffad6e8ed4449dafe11b82c7aea82f74,expr0,M,all,1202
ffad6e8ed4449dafe11b82c7aea82f74,expr0,H,M,53.2
ffad6e8ed4449dafe11b82c7aea82f74,expr0,H,F,64
ffad6e8ed4449dafe11b82c7aea82f74,expr0,H,all,1203
ffad6e8ed4449dafe11b82c7aea82f74,expr1,L,M,6
ffad6e8ed4449dafe11b82c7aea82f74,expr1,L,F,10
ffad6e8ed4449dafe11b82c7aea82f74,expr1,L,all,3206
ffad6e8ed4449dafe11b82c7aea82f74,expr1,M,M,10
ffad6e8ed4449dafe11b82c7aea82f74,expr1,M,F,14
ffad6e8ed4449dafe11b82c7aea82f74,expr1,M,all,3210
ffad6e8ed4449dafe11b82c7aea82f74,expr1,H,M,14
ffad6e8ed4449dafe11b82c7aea82f74,expr1,H,F,18
ffad6e8ed4449dafe11b82c7aea82f74,expr1,H,all,3214
ffad6e8ed4449dafe11b82c7aea82f74,expr2,L,M,51.5
ffad6e8ed4449dafe11b82c7aea82f74,expr2,L,F,62.5
ffad6e8ed4449dafe11b82c7aea82f74,expr2,L,all,2002.5
ffad6e8ed4449dafe11b82c7aea82f74,expr2,M,M,54.1
ffad6e8ed4449dafe11b82c7aea82f74,expr2,M,F,65.5
ffad6e8ed4449dafe11b82c7aea82f74,expr2,M,all,2004.5
ffad6e8ed4449dafe11b82c7aea82f74,expr2,H,M,56.7
ffad6e8ed4449dafe11b82c7aea82f74,expr2,H,F,68.5
ffad6e8ed4449dafe11b82c7aea82f74,expr2,H,all,2006.5
ffad6e8ed4449dafe11b82c7aea82f74,expr3,L,M,75
ffad6e8ed4449dafe11b82c7aea82f74,expr3,L,F,150
ffad6e8ed4449dafe11b82c7aea82f74,expr3,L,all,963601.5
ffad6e8ed4449dafe11b82c7aea82f74,expr3,M,M,129
ffad6e8ed4449dafe11b82c7aea82f74,expr3,M,F,217
ffad6e8ed4449dafe11b82c7aea82f74,expr3,M,all,965605
ffad6e8ed4449dafe11b82c7aea82f74,expr3,H,M,186.2
ffad6e8ed4449dafe11b82c7aea82f74,expr3,H,F,288
ffad6e8ed4449dafe11b82c7aea82f74,expr3,H,all,967610.5
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr0,L,M,1
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr0,L,F,1
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr0,L,all,1201
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr0,M,M,1
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr0,M,F,1
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr0,M,all,601
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr0,H,M,1
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr0,H,F,1
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr0,H,all,401
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr1,L,M,null
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr1,L,F,10
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr1,L,all,4.0075
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr1,M,M,10
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr1,M,F,7
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr1,M,all,4.00749063670412
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr1,H,M,7
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr1,H,F,6
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr1,H,all,4.0074812967581
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr2,L,M,1.03
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr2,L,F,1.02459016393443
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr2,L,all,2.5
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr2,M,M,1.02851711026616
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr2,M,F,1.0234375
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr2,M,all,2.49626400996264
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr2,H,M,1.02717391304348
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr2,H,F,1.02238805970149
```

```
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr2,H,all,2.49254658385093
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr3,L,M,null
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr3,L,F,2.5
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr3,L,all,1204.501875
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr3,M,M,2.5
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr3,M,F,1.75
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr3,M,all,602.749687890137
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr3,H,M,1.75
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr3,H,F,1.5
ffad6e8ed4449dafe11b82c7aea82f74,ratio_expr3,H,all,402.165627597672
```

GET csv model runs comparison table expressions (enum id's)

Compare model runs and return results as csv file.

Compare `[base]` and `[variant]` model runs output values for each expression and get it as response stream UTF-8 outputTable.csv file attachment, optionally starts with byte order mark (BOM).

Comparison can be calculated as one of the following:

- for each table expression use one of: `diff`, `ratio` or `percent` comparison between `[base]` and `[variant]` model runs.
- use comma separated list of comparison expressions between `[base]` and `[variant]` or simple expression for each run.

Dimension(s) returned as enum id's.

Methods:

```
GET /api/model/:model/run/:run/table/:name/compare/:compare/variant/:variant/csv-id
GET /api/model/:model/run/:run/table/:name/compare/:compare/variant/:variant/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) base model run digest, run stamp or run name
:variant - (required) variant model run(s): comma-separated list of digests, run stamps or run names
```

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
```

```
:compare - (required) comparison to calculate
```

- `diff` differenece of values between variant and base run, e.g.: `Expr0[variant] - Expr0[base]`
- `ratio` ratio of values between variant and base run, e.g.: `Expr0[variant] / Expr0[base]`
- `percent` proportional difference multipled by 100, e.g.: `100 * (Expr0[variant] - Expr0[base]) / Expr0[base]`

Or a list of comma-separated expressions, for example: `expr0, expr1[variant] + expr2[base]`

It is also possible to use parameter(s) in calculation, parameter must be a scalar of float or integer type. For example: `(Expr1[variant] - Expr1[base]) * param.Scale[base]`, where `param.Scale` is a value of scalar parameter `Scale` in `[base]` model run.

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/compare/diff/variant/Default-4/csv
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/compare/ratio/variant/Default-4/csv
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/compare/percent/variant/Default-4/csv

http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/compare/diff/variant/Default-4/csv-bom

http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/compare/expr0,expr1,expr2/variant/Default-4/csv
http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/compare/expr0%2Cexpr1%5Bvariant%5D%2Bexpr2%5Bbase%5D/variant/Default-4/csv
```

Note: above `expr0%2Cexpr1%5Bvariant%5D%2Bexpr2%5Bbase%5D` is URL encoded: `expr0,expr1[variant]+expr2[base]`.

Example:

In output below `CalcId` contains output table expression id's: `0 <= ExprId < 12000` or id's of calculated values: `ExprId + 12000`. For example, id of calculated value `Expr2[variant] - Expr2[base]` is: `CalcId = 12002`

```
curl http://localhost:4040/api/model/modelOne/run/Default/table/salarySex/compare/ratio/variant/Default-4,Sub-values_4/csv-id
```

```
run_id,calc_id,dim0,dim1,calc_value
201,0,100,0,50
201,0,100,1,60
201,0,100,800,1
201,0,200,0,51.6
201,0,200,1,62
201,0,200,800,2
201,0,300,0,53.2
201,0,300,1,64
201,0,300,800,3
201,1,100,0,0
201,1,100,1,1
201,1,100,800,800
201,1,200,0,1
201,1,200,1,2
201,1,200,800,801
201,1,300,0,2
201,1,300,1,3
201,1,300,800,802
201,2,100,0,50
201,2,100,1,61
201,2,100,800,801
201,2,200,0,52.6
201,2,200,1,64
201,2,200,800,803
201,2,300,0,55.2
201,2,300,1,67
201,2,300,800,805
201,3,100,0,0
201,3,100,1,60
201,3,100,800,800
201,3,200,0,51.6
201,3,200,1,124
201,3,200,800,1602
201,3,300,0,106.4
201,3,300,1,192
201,3,300,800,2406
202,0,100,0,50
202,0,100,1,60
202,0,100,800,1201
202,0,200,0,51.6
202,0,200,1,62
202,0,200,800,1202
202,0,300,0,53.2
202,0,300,1,64
202,0,300,800,1203
202,1,100,0,6
202,1,100,1,10
202,1,100,800,3206
202,1,200,0,10
202,1,200,1,14
202,1,200,800,3210
202,1,300,0,14
202,1,300,1,18
202,1,300,800,3214
202,2,100,0,51.5
202,2,100,1,62.5
202,2,100,800,2002.5
202,2,200,0,54.1
202,2,200,1,65.5
202,2,200,800,2004.5
202,2,300,0,56.7
202,2,300,1,68.5
202,2,300,800,2006.5
202,3,100,0,75
202,3,100,1,150
202,3,100,800,963601.5
202,3,200,0,129
202,3,200,1,217
202,3,200,800,965605
202,3,300,0,186.2
202,3,300,1,288
202,3,300,800,967610.5
202,12000,100,0,1
202,12000,100,1,1
202,12000,100,800,1201
202,12000,200,0,1
202,12000,200,1,1
202,12000,200,800,601
```

202,12000,300,0,1
202,12000,300,1,1
202,12000,300,800,401
202,12001,100,0,null
202,12001,100,1,10
202,12001,100,800,4.0075
202,12001,200,0,10
202,12001,200,1,7
202,12001,200,800,4.00749063670412
202,12001,300,0,7
202,12001,300,1,6
202,12001,300,800,4.0074812967581
202,12002,100,0,1.03
202,12002,100,1,1.02459016393443
202,12002,100,800,2.5
202,12002,200,0,1.02851711026616
202,12002,200,1,1.0234375
202,12002,200,800,2.49626400996264
202,12002,300,0,1.02717391304348
202,12002,300,1,1.02238805970149
202,12002,300,800,2.49254658385093
202,12003,100,0,null
202,12003,100,1,2.5
202,12003,100,800,1204.501875
202,12003,200,0,2.5
202,12003,200,1,1.75
202,12003,200,800,602.749687890137
202,12003,300,0,1.75
202,12003,300,1,1.5
202,12003,300,800,402.165627597672
208,0,100,0,50
208,0,100,1,60
208,0,100,800,1201
208,0,200,0,51.6
208,0,200,1,62
208,0,200,800,1202
208,0,300,0,53.2
208,0,300,1,64
208,0,300,800,1203
208,1,100,0,6
208,1,100,1,10
208,1,100,800,3206
208,1,200,0,10
208,1,200,1,14
208,1,200,800,3210
208,1,300,0,14
208,1,300,1,18
208,1,300,800,3214
208,2,100,0,51.5
208,2,100,1,62.5
208,2,100,800,2002.5
208,2,200,0,54.1
208,2,200,1,65.5
208,2,200,800,2004.5
208,2,300,0,56.7
208,2,300,1,68.5
208,2,300,800,2006.5
208,3,100,0,75
208,3,100,1,150
208,3,100,800,963601.5
208,3,200,0,129
208,3,200,1,217
208,3,200,800,965605
208,3,300,0,186.2
208,3,300,1,288
208,3,300,800,967610.5
208,12000,100,0,1
208,12000,100,1,1
208,12000,100,800,1201
208,12000,200,0,1
208,12000,200,1,1
208,12000,200,800,601
208,12000,300,0,1
208,12000,300,1,1
208,12000,300,800,401
208,12001,100,0,null
208,12001,100,1,10
208,12001,100,800,4.0075
208,12001,200,0,10
208,12001,200,1,7
208,12001,200,800,4.00749063670412
208,12001,300,0,7
208,12001,300,1,6
208,12001,300,800,4.0074812967581
208,12002,100,0,1.03
208,12002,100,1,1.02459016393443
208,12002,100,800,2.5
208,12002,200,0,1.02851711026616

```
208,12002,200,1,1.0234375
208,12002,200,800,2.49626400996264
208,12002,300,0,1.02717391304348
208,12002,300,1,1.02238805970149
208,12002,300,800,2.49254658385093
208,12003,100,0,null
208,12003,100,1,2.5
208,12003,100,800,1204.501875
208,12003,200,0,2.5
208,12003,200,1,1.75
208,12003,200,800,602.749687890137
208,12003,300,0,1.75
208,12003,300,1,1.5
208,12003,300,800,402.165627597672
```

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 csv aggregated microdata from model run

Aggregate microdata values and read it as csv file.

Aggregate microdata value attributes, group by dimension attributes and get it as response stream UTF-8 Entity.csv file attachment, optionally starts with byte order mark (BOM).

Result can include multiple aggregations of value attributes (float of integer type) and group by dimension attributes (enum-based or bool type). Aggregation(s) is a comma-separated list of Model Output Expressions of microdata value attributes. For example, two aggregations: `OM_AVG(Income) , OM_MAX(Salary + Pension)` and group by two dimension attributes: `AgeGroup , Sex`.

It is also possible to use parameter(s) in calculation, parameter must be a scalar of float or integer type. For example: `OM_COUNT_IF(Income > param.High)`, where `param.High` is a value of scalar parameter `High` in that model run.

Enum-based microdata attributes returned as enum codes.

Following aggregation functions avaliable:

- `OM_AVG` mean of accumulators sub-values
- `OM_SUM` sum of accumulators sub-values
- `OM_COUNT` count of accumulators sub-values (excluding NULL's)
- `OM_COUNT_IF` count values matching condition
- `OM_MAX` maximum of accumulators sub-values
- `OM_MIN` minimum of accumulators sub-values
- `OM_VAR` variance of accumulators sub-values
- `OM_SD` standard deviation of accumulators sub-values
- `OM_SE` standard error of accumulators sub-values
- `OM_CV` coefficient of variation of accumulators sub-values

For more details please see: Model Output Expressions

Methods:

```
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/calc/:calc/csv
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/calc/:calc/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
```

```
:group-by - (required) comma-separated list of dimension attribute(s) to group by aggreagated values, dimension attribute must be enum-based or boolean type.
```

```
:calc - (required) comma-separated list of aggregation of microdata value attribute(s), value attribute must be float of integer type.
```

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/calc/OM_AVG(Income)/csv
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/calc/OM_AVG(Income)/csv-bom
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/calc/OM_AVG(Income),OM_AVG(Salary+Pension)/csv
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/calc/OM_AVG(Income),OM_AVG(Salary+Pension)/csv-bo
m
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/calc/OM_COUNT_IF(Income%3Eparam.StartingSeed)/csv
```

Note: `OM_COUNT_IF(Income%3Eparam.StartingSeed)` is is URL encoded: `OM_COUNT_IF(Income>param.StartingSeed)`

Return example:

```
curl http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/calc/OM_AVG(Income)/csv
```

```
run_digest,calc_name,AgeGroup,Sex,calc_value
a59c91359c4cd98f6275529c798d2485,ex_12000,10-20,M,13400876.3543608
a59c91359c4cd98f6275529c798d2485,ex_12000,10-20,F,13403741.8890926
a59c91359c4cd98f6275529c798d2485,ex_12000,20-30,M,134201344
a59c91359c4cd98f6275529c798d2485,ex_12000,20-30,F,134209535
a59c91359c4cd98f6275529c798d2485,ex_12000,30-40,M,134283254
a59c91359c4cd98f6275529c798d2485,ex_12000,30-40,F,134291445
a59c91359c4cd98f6275529c798d2485,ex_12000,40+,M,74645804.26116
a59c91359c4cd98f6275529c798d2485,ex_12000,40+,F,71069306.5718732
```

```
curl http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/calc/OM_AVG(Income),OM_AVG(Salary+Pension)/csv
```

```
run_digest,calc_name,AgeGroup,Sex,calc_value
a59c91359c4cd98f6275529c798d2485,ex_12000,10-20,M,13400876.3543608
a59c91359c4cd98f6275529c798d2485,ex_12000,10-20,F,13403741.8890926
a59c91359c4cd98f6275529c798d2485,ex_12000,20-30,M,134201344
a59c91359c4cd98f6275529c798d2485,ex_12000,20-30,F,134209535
a59c91359c4cd98f6275529c798d2485,ex_12000,30-40,M,134283254
a59c91359c4cd98f6275529c798d2485,ex_12000,30-40,F,134291445
a59c91359c4cd98f6275529c798d2485,ex_12000,40+,M,74645804.26116
a59c91359c4cd98f6275529c798d2485,ex_12000,40+,F,71069306.5718732
a59c91359c4cd98f6275529c798d2485,ex_12001,10-20,M,10050657.2657706
a59c91359c4cd98f6275529c798d2485,ex_12001,10-20,F,10052806.4168194
a59c91359c4cd98f6275529c798d2485,ex_12001,20-30,M,100651008
a59c91359c4cd98f6275529c798d2485,ex_12001,20-30,F,100657151.25
a59c91359c4cd98f6275529c798d2485,ex_12001,30-40,M,100712440.5
a59c91359c4cd98f6275529c798d2485,ex_12001,30-40,F,100718583.75
a59c91359c4cd98f6275529c798d2485,ex_12001,40+,M,60124633.8262087
a59c91359c4cd98f6275529c798d2485,ex_12001,40+,F,57665830.5421598
```

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

Aggregate microdata values and read it as csv file.

Aggregate microdata value attributes, group by dimension attributes and get it as response stream UTF-8 Entity.csv file attachment, optionally starts with byte order mark (BOM).

Result can include multiple aggregations of value attributes (float of integer type) and group by dimension attributes (enum-based or bool type). Aggregation(s) is a comma-separated list of Model Output Expressions of microdata value attributes. For example, two aggregations: `OM_AVG(Income) , OM_MAX(Salary + Pension)` and group by two dimension attributes: `AgeGroup , Sex`.

It is also possible to use parameter(s) in calculation, parameter must be a scalar of float or integer type. For example: `OM_COUNT_IF(Income > param.High)`, where `param.High` is a value of scalar parameter `High` in that model run.

Enum-based microdata attributes returned as enum id, not enum codes.

Following aggregation functions avaliable:

- `OM_AVG` mean of accumulators sub-values
- `OM_SUM` sum of accumulators sub-values
- `OM_COUNT` count of accumulators sub-values (excluding NULL's)
- `OM_COUNT_IF` count values matching condition
- `OM_MAX` maximum of accumulators sub-values
- `OM_MIN` minimum of accumulators sub-values
- `OM_VAR` variance of accumulators sub-values
- `OM_SD` standard deviation of accumulators sub-values
- `OM_SE` standard error of accumulators sub-values
- `OM_CV` coefficient of variation of accumulators sub-values

For more details please see: Model Output Expressions

Methods:

```
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/calc/:calc/csv-id
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/calc/:calc/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
```

```
:group-by - (required) comma-separated list of dimension attribute(s) to group by aggreagated values, dimension attribute must be enum-based or boolean type.
```

```
:calc - (required) comma-separated list of aggregation of microdata value attribute(s), value attribute must be float of integer type.
```

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/calc/OM_AVG(Income)/csv-id
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/calc/OM_AVG(Income)/csv-id-bom
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/calc/OM_AVG(Income),OM_AVG(Salary+Pension)/csv-id
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/calc/OM_AVG(Income),OM_AVG(Salary+Pension)/csv-id-bom
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/calc/OM_COUNT_IF(Income%3Eparam.StartingSeed)/csv-id
```

Note: `OM_COUNT_IF(Income%3Eparam.StartingSeed)` is is URL encoded: `OM_COUNT_IF(Income>param.StartingSeed)`

Return example:

```
curl http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/calc/OM_AVG(Income)/csv-id
```

```
run_id,calc_id,AgeGroup,Sex,calc_value
219,12000,10,0,13400876.3543608
219,12000,10,1,13403741.8890926
219,12000,20,0,134201344
219,12000,20,1,134209535
219,12000,30,0,134283254
219,12000,30,1,134291445
219,12000,40,0,74645804.26116
219,12000,40,1,71069306.5718732
```

```
curl http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/calc/OM_AVG(Income),OM_AVG(Salary+Pension)/csv-id
```

```
run_id,calc_id,AgeGroup,Sex,calc_value
219,12000,10,0,13400876.3543608
219,12000,10,1,13403741.8890926
219,12000,20,0,134201344
219,12000,20,1,134209535
219,12000,30,0,134283254
219,12000,30,1,134291445
219,12000,40,0,74645804.26116
219,12000,40,1,71069306.5718732
219,12001,10,0,10050657.2657706
219,12001,10,1,10052806.4168194
219,12001,20,0,100651008
219,12001,20,1,100657151.25
219,12001,30,0,100712440.5
219,12001,30,1,100718583.75
219,12001,40,0,60124633.8262087
219,12001,40,1,57665830.5421598
```

GET csv microdata run comparison

Compare microdata values and return results as csv file.

Compare `[base]` and `[variant]` model runs microdata value attributes (float of integer type), group it by dimension attributes (enum-based or bool type) and get it as response stream UTF-8 Entity.csv file attachment, optionally starts with byte order mark (BOM).

Result can include multiple aggregated comparisons, grouped by multiple dimension attributes. Aggregated comparision(s) is a comma-separated list of Model Output Expressions of `[base]` and `[variant]` value attributes. For example, two comparisions: `OM_AVG(Income[variant] - Income[base]) , OM_MAX( 100 * (Salary[variant] + Pension[variant]) / Income[base])` and group by two dimension attributes: `AgeGroup , Sex`.

It is also possible to use parameter(s) in calculation, parameter must be a scalar of float or integer type. For example: `OM_COUNT_IF((Income[variant - Income[base]) > param.High[base])`, where `param.High[base]` is a value of scalar parameter `High` in `[base]` model run.

Enum-based microdata attributes returned as enum codes.

Following aggregation functions avaliable:

- `OM_AVG` mean of accumulators sub-values
- `OM_SUM` sum of accumulators sub-values
- `OM_COUNT` count of accumulators sub-values (excluding NULL's)
- `OM_COUNT_IF` count values matching condition
- `OM_MAX` maximum of accumulators sub-values
- `OM_MIN` minimum of accumulators sub-values
- `OM_VAR` variance of accumulators sub-values
- `OM_SD` standard deviation of accumulators sub-values
- `OM_SE` standard error of accumulators sub-values
- `OM_CV` coefficient of variation of accumulators sub-values

For more details please see: Model Output Expressions

Methods:

```
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/compare/:compare/variant/:variant/csv
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/compare/:compare/variant/:variant/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) base model run digest, run stamp or run name
:variant - (required) variant model run(s): comma-separated list of digests, run stamps or run names
```

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
```

```
:group-by - (required) comma-separated list of dimension attribute(s) to group by aggreagated values, dimension attribute must be enum-based or boolean type.
```

:compare - (required) comma-separated list of comparions of microdata value attribute(s), value attribute must be float of integer type.

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/compare/OM_AVG(Income%5Bvariant%5D-Income%5Bbase%5D)/variant/Microdata%20other%20in%20database/csv
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/compare/OM_AVG(Income%5Bvariant%5D-Income%5Bbase%5D)/variant/Microdata%20other%20in%20database/csv-bom

http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/compare/OM_AVG(Income%5Bvariant%5D-Income%5Bbase%5D),OM_AVG(Salary)/variant/Microdata%20other%20in%20database/csv
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/compare/OM_AVG((Income%5Bvariant%5D-Income%5Bbase%5D)%2Aparam.StartingSeed%5Bbase%5D)/variant/Microdata%20other%20in%20database/csv
```

Note:

- `OM_AVG(Income%5Bvariant%5D-Income%5Bbase%5D)` is URL encoded: `OM_AVG(Income[variant]-Income[base])`
- `OM_AVG((Income%5Bvariant%5D-Income%5Bbase%5D)%2Aparam.StartingSeed%5Bbase%5D)` is URL encoded: `OM_AVG((Income[variant]-Income[base])*param.StartingSeed)`

Return example:

```
curl http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/compare/OM_AVG(Income%5Bvariant%5D-Income%5Bbase%5D)/variant/Microdata%20other%20in%20database/csv
```

```
run_digest,calc_name,AgeGroup,Sex,calc_value
86135ceed94d1239937a42e088a7fcb7,ex_12000,10-20,M,-6701256.20161906
86135ceed94d1239937a42e088a7fcb7,ex_12000,10-20,F,-6702689.14390467
86135ceed94d1239937a42e088a7fcb7,ex_12000,20-30,M,-67108864
86135ceed94d1239937a42e088a7fcb7,ex_12000,20-30,F,-67112960
86135ceed94d1239937a42e088a7fcb7,ex_12000,30-40,M,-67149824
86135ceed94d1239937a42e088a7fcb7,ex_12000,30-40,F,-67153920
86135ceed94d1239937a42e088a7fcb7,ex_12000,40+,M,-37327458.4707197
86135ceed94d1239937a42e088a7fcb7,ex_12000,40+,F,-35538991.0909255
```

```
curl http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/compare/OM_AVG(Income%5Bvariant%5D-Income%5Bbase%5D),OM_AVG(Salary)/variant/Microdata%20other%20in%20database/csv
```

```
run_digest,calc_name,AgeGroup,Sex,calc_value
a59c91359c4cd98f6275529c798d2485,ex_12001,10-20,M,10050657.2657706
a59c91359c4cd98f6275529c798d2485,ex_12001,10-20,F,10052806.4168194
a59c91359c4cd98f6275529c798d2485,ex_12001,20-30,M,100651008
a59c91359c4cd98f6275529c798d2485,ex_12001,20-30,F,100657151.25
a59c91359c4cd98f6275529c798d2485,ex_12001,30-40,M,100712440.5
a59c91359c4cd98f6275529c798d2485,ex_12001,30-40,F,100718583.75
a59c91359c4cd98f6275529c798d2485,ex_12001,40+,M,44905401.6613024
a59c91359c4cd98f6275529c798d2485,ex_12001,40+,F,41552168.3348642
86135ceed94d1239937a42e088a7fcb7,ex_12000,10-20,M,-6701256.20161906
86135ceed94d1239937a42e088a7fcb7,ex_12000,10-20,F,-6702689.14390467
86135ceed94d1239937a42e088a7fcb7,ex_12000,20-30,M,-67108864
86135ceed94d1239937a42e088a7fcb7,ex_12000,20-30,F,-67112960
86135ceed94d1239937a42e088a7fcb7,ex_12000,30-40,M,-67149824
86135ceed94d1239937a42e088a7fcb7,ex_12000,30-40,F,-67153920
86135ceed94d1239937a42e088a7fcb7,ex_12000,40+,M,-37327458.4707197
86135ceed94d1239937a42e088a7fcb7,ex_12000,40+,F,-35538991.0909255
86135ceed94d1239937a42e088a7fcb7,ex_12001,10-20,M,5024715.11455629
86135ceed94d1239937a42e088a7fcb7,ex_12001,10-20,F,5025789.55889093
86135ceed94d1239937a42e088a7fcb7,ex_12001,20-30,M,50319360
86135ceed94d1239937a42e088a7fcb7,ex_12001,20-30,F,50322431.25
86135ceed94d1239937a42e088a7fcb7,ex_12001,30-40,M,50350072.5
86135ceed94d1239937a42e088a7fcb7,ex_12001,30-40,F,50353143.75
86135ceed94d1239937a42e088a7fcb7,ex_12001,40+,M,22449959.7419271
86135ceed94d1239937a42e088a7fcb7,ex_12001,40+,F,20773547.8187544
```

GET csv microdata run comparison (enum id's)

Compare microdata values and return results as csv file.

Compare `[base]` and `[variant]` model runs microdata value attributes (float of integer type), group it by dimension attributes (enum-based or bool type) and get it as response stream UTF-8 Entity.csv file attachment, optionally starts with byte order mark (BOM).

Result can include multiple aggregated comparisons, grouped by multiple dimension attributes. Aggregated comparision(s) is a comma-separated list of Model Output Expressions of `[base]` and `[variant]` value attributes. For example, two comparisions: `OM_AVG(Income[variant] - Income[base]) , OM_MAX( 100 * (Salary[variant] + Pension[variant]) / Income[base])` and group by two dimension attributes: `AgeGroup , Sex`.

It is also possible to use parameter(s) in calculation, parameter must be a scalar of float or integer type. For example: `OM_COUNT_IF((Income[variant - Income[base]) > param.High[base])`, where `param.High[base]` is a value of scalar parameter `High` in `[base]` model run.

Enum-based microdata attributes returned as enum id, not enum codes.

Following aggregation functions avaliable:

- `OM_AVG` mean of accumulators sub-values
- `OM_SUM` sum of accumulators sub-values
- `OM_COUNT` count of accumulators sub-values (excluding NULL's)
- `OM_COUNT_IF` count values matching condition
- `OM_MAX` maximum of accumulators sub-values
- `OM_MIN` minimum of accumulators sub-values
- `OM_VAR` variance of accumulators sub-values
- `OM_SD` standard deviation of accumulators sub-values
- `OM_SE` standard error of accumulators sub-values
- `OM_CV` coefficient of variation of accumulators sub-values

For more details please see: Model Output Expressions

Methods:

```
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/compare/:compare/variant/:variant/csv-id
GET /api/model/:model/run/:run/microdata/:name/group-by/:group-by/compare/:compare/variant/:variant/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) base model run digest, run stamp or run name
:variant - (required) variant model run(s): comma-separated list of digests, run stamps or run names
```

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
```

```
:group-by - (required) comma-separated list of dimension attribute(s) to group by aggreagated values, dimension attribute must be enum-based or boolean type.
```

:compare - (required) comma-separated list of comparions of microdata value attribute(s), value attribute must be float of integer type.

Call examples:

```
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/compare/OM_AVG(Income%5Bvariant%5D-Income%5Bbase%5D)/variant/Microdata%20other%20in%20database/csv-id
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/compare/OM_AVG(Income%5Bvariant%5D-Income%5Bbase%5D)/variant/Microdata%20other%20in%20database/csv-id-bom

http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/compare/OM_AVG(Income%5Bvariant%5D-Income%5Bbase%5D),OM_AVG(Salary)/variant/Microdata%20other%20in%20database/csv-id
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/compare/OM_AVG(Income%5Bvariant%5D-Income%5Bbase%5D),OM_AVG(Salary)/variant/Microdata%20other%20in%20database/csv-id-bom
http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/compare/OM_AVG((Income%5Bvariant%5D-Income%5Bbase%5D)%2Aparam.StartingSeed%5Bbase%5D)/variant/Microdata%20other%20in%20database/csv-id
```

Note:

- `OM_AVG(Income%5Bvariant%5D-Income%5Bbase%5D)` is URL encoded: `OM_AVG(Income[variant]-Income[base])`
- `OM_AVG((Income%5Bvariant%5D-Income%5Bbase%5D)%2Aparam.StartingSeed%5Bbase%5D)` is URL encoded: `OM_AVG((Income[variant]-Income[base])*param.StartingSeed)[base]`

Return example:

```
curl http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/compare/OM_AVG(Income%5Bvariant%5D-Income%5Bbase%5D)/variant/Microdata%20other%20in%20database/csv-id
```

```
run_id,calc_id,AgeGroup,Sex,calc_value
221,12000,10,0,-6701256.20161906
221,12000,10,1,-6702689.14390467
221,12000,20,0,-67108864
221,12000,20,1,-67112960
221,12000,30,0,-67149824
221,12000,30,1,-67153920
221,12000,40,0,-37327458.4707197
221,12000,40,1,-35538991.0909255
```

```
curl http://localhost:4040/api/model/modelOne/run/Microdata%20in%20database/microdata/Person/group-by/AgeGroup,Sex/compare/OM_AVG(Income%5Bvariant%5D-Income%5Bbase%5D),OM_AVG(Salary)/variant/Microdata%20other%20in%20database/csv-id
```

```
run_id,calc_id,AgeGroup,Sex,calc_value
219,12001,10,0,10050657.2657706
219,12001,10,1,10052806.4168194
219,12001,20,0,100651008
219,12001,20,1,100657151.25
219,12001,30,0,100712440.5
219,12001,30,1,100718583.75
219,12001,40,0,44905401.6613024
219,12001,40,1,41552168.3348642
221,12000,10,0,-6701256.20161906
221,12000,10,1,-6702689.14390467
221,12000,20,0,-67108864
221,12000,20,1,-67112960
221,12000,30,0,-67149824
221,12000,30,1,-67153920
221,12000,40,0,-37327458.4707197
221,12000,40,1,-35538991.0909255
221,12001,10,0,5024715.11455629
221,12001,10,1,5025789.55889093
221,12001,20,0,50319360
221,12001,20,1,50322431.25
221,12001,30,0,50350072.5
221,12001,30,1,50353143.75
221,12001,40,0,22449959.7419271
221,12001,40,1,20773547.8187544
```

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
```

POST delete multiple worksets

Delete multiple worksets 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:

```
POST /api/model/:model/delete-worksets
```

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.

JSON body request:

Array of workset names to delete, for example:

```
[
  "modelOne_partial",
  "modelOne_other"
]
```

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 example:

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/delete-worksets -d @del_m1_ws_lst.json

Note: Unnecessary use of -X or --request, POST is already inferred.
*   Trying [::1]:4040...
* Connected to localhost (::1) port 4040
> POST /api/model/modelOne/delete-worksets HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/8.4.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 96
>
< HTTP/1.1 200 OK
< Content-Location: /api/model/modelOne//delete-worksets/2
< Content-Type: text/plain
< Date: Wed, 24 Jan 2024 04:49:52 GMT
< Content-Length: 0
<
* 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/_20120817160459014 8_/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": "_20120817160459014 8_",
  "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, input parameters and microdata.

Method:

```
DELETE /api/model/:model/run/:run
```

This method is asynchronous: it starts run delete return a response immediately. You need to check the results later by using GET status of model run or GET list of model runs or other methods to make sure that particular model run deleted and no longer exist in the list of model 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.

```
: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/unlink/run/Default-4
```

```
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
```

POST delete model runs

Delete multiple model runs from database, including output table values, input parameters and microdata.

Method:

```
POST /api/model/:model/delete-runs
```

This method is asynchronous: it starts runs delete return a response immediately. You need to check the results later by using GET list of model runs or other methods to make sure model runs are deleted and no longer exist in the list of model 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.

JSON body request:

Array of model run digests, run stamps or run names, for example:

```
[
  "ae4546791da1b828a4fe0c0ca0935e7a",
  "2023_12_29_23_46_19_255",
  "First Task Run_modelOne_other",
  "Task Run with Suppressed Tables_modelOne_other"
]
```

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 example:

```
curl -v -X POST -H "Content-Type: application/json" http://localhost:4040/api/model/modelOne/delete-runs -d @del_m1_runs.json

Note: Unnecessary use of -X or --request, POST is already inferred.
*   Trying [::1]:4040...
* Connected to localhost (::1) port 4040
> POST /api/model/modelOne/delete-runs HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/8.4.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 172
>
< HTTP/1.1 200 OK
< Content-Location: /api/model/modelOne/delete-runs/4
< Content-Type: text/plain
< Date: Wed, 24 Jan 2024 04:02:54 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.js
on
```

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`.

INI file path

You can use `OpenM.IniFile` model run option to specify path to model run ini-file. By default ini file has a name: `ModelName.ini` and it is located in the same directory where model executable is.

If you wnat to use ini files from your server User Files Directory then please start your ini file path with `OM_USER_FILES` prefix. For example:

```
{
  "ModelName":       "modelOne",
  "ModelDigest":     "_201208171604590148_",
  "Opts": {
    "OpenM.IniFile": "OM_USER_FILES/my.ini"
  }
}
```

Example 1:

Run modelOne.exe with 2 sub-values (sub-value is similar to Modgen "sub-sample"):

```
{
  "ModelName": "modelOne",
  "Opts": {
    "OpenM.SubValues": "2"
  }
}
```

Important: `Opts` values must be a "quoted string". In above JSON number of sub-values is `"2"` and not `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 1000 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, if count <= 0 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,
```

"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 allow downloads by user.

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

Methods:

```
DELETE /api/download/delete/:folder
DELETE /api/download/start/delete/:folder
```

First method deletes download files from the server. Second method will initiate deleting of download files from the server. Actual delete performed in background and results can be checked through log file.

Arguments:

```
:folder - (required) download folder file name, for example: `RiskPaths.run.102`
```

Call examples:

```
curl -v -X DELETE http://localhost:4040/api/download/delete/RiskPaths.run.102
curl -v -X DELETE http://localhost:4040/api/download/start/delete/RiskPaths.run.102
```

DELETE all download files

DELETE all download files from the server.

This method only avaliable if server configured to allow downloads by user.

It does delete all files from server download folder.

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

Methods:

```
DELETE /api/download/delete-all
DELETE /api/download/start/delete-all
```

First method deletes all files from server download folder. Second method will initiate deleting the files. Actual delete performed in background and user may need to refresh page to verify the results.

Call examples:

```
curl -v -X DELETE http://localhost:4040/api/download/delete-all
curl -v -X DELETE http://localhost:4040/api/download/start/delete-all
```

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\\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. ",
    ""
  ]
}

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\\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. ",
      ""
    ]
  },
  {
    "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.InputD
ir 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 allow uploads to the server by user.

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

Methods:

```
DELETE /api/upload/delete/:folder
DELETE /api/upload/start/delete/:folder
```

First method deletes upload files from the server.

Second method will initiate deleting of upload files from the server. Actual delete performed in background and results can be checked through log file.

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
curl -v -X DELETE http://localhost:4040/api/upload/start/delete/RiskPaths.set.New-Data
```

DELETE all upload files

DELETE all upload files from the server.

This method only avaliable if server configured to allow uploads to the server by user.

It does delete all files from server download folder.

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

Methods:

```
DELETE /api/upload/delete-all
DELETE /api/upload/start/delete-all
```

First method deletes all files from server upload folder.

Second method will initiate deleting the files. Actual delete performed in background and results can be checked through log file.

Call example:

```
curl -v -X DELETE http://localhost:4040/api/upload/delete-all
curl -v -X DELETE http://localhost:4040/api/upload/start/delete-all
```

GET user files tree

GET files tree from user files directory on the server.

If user files access enabled in server configuration then user can upload or download files, create or delete folders in user files directory.

This is a beta version and may change in the future.

This method retruns files and folders tree under user files server directory.

Method:

```
GET /api/files/file-tree/:ext/path/:path
GET /api/files/file-tree/:ext/path/
GET /api/files/file-tree/:ext/path
GET /api/files/file-tree/:ext/path?path=....
```

Arguments:

```
:ext -  (required) comma separated list of files extension, for example: `ini` or `csv,tsv`. Use `_  or `*` to get all files, with any extension.
:path - (optional) if not empty then return list of the files in specified folder, else entire tree of user files.
```

Call examples from browser:

```
http://localhost:4040/api/files/file-tree/ini/path
http://localhost:4040/api/files/file-tree/.INI/path
http://localhost:4040/api/files/file-tree/csv%2Ctsv/path/upload
http://localhost:4040/api/files/file-tree/_/path
http://localhost:4040/api/files/file-tree/log/path/download
http://localhost:4040/api/files/file-tree/_/path?path=upload%2FmodelOne.set.New
```

Please notice URL encoded: `csv%2Ctsv` which is `csv,tsv` and `?path=upload%2FmodelOne.set.New` is used to specify `upload/modelOne.set.New` `path` query parameter.

Return example:


```
[
  {
    "Path": "upload/modelOne.set.New",
    "IsDir": true,
    "Size": 0,
    "ModTime": 1718249290709
  },
  {
    "Path": "upload/modelOne.set.New/modelOne.set.New",
    "IsDir": true,
    "Size": 0,
    "ModTime": 1718249290709
  },
  {
    "Path": "upload/modelOne.set.New/modelOne.set.New/modelOne.set.New.json",
    "IsDir": false,
    "Size": 267,
    "ModTime": 1718249290709
  },
  {
    "Path": "upload/modelOne.set.New/modelOne.set.New/set.New",
    "IsDir": true,
    "Size": 0,
    "ModTime": 1718249290709
  }
]
```

POST upload to user files

POST upload file to user files directory on the server.

If user files access enabled in server configuration then user can upload or download files, create or delete folders in user files directory.

This is a beta version and may change in the future.

User can upload any file into user files directory or sub-directories. If this is a `.zip` archive then it will be automatically unpacked.

Method:

```
POST /api/files/file/:path
POST /api/files/file?path=....
```

Arguments:

```
:path - (required) directory and file name to be uploaded.
```

Call examples:

```
curl -v -X POST -F "filename=@a.ini" http://localhost:4040/api/files/file/a.ini
curl -v -X POST -F "filename=@a.ini" http://localhost:4040/api/files/file?path=dir%2Fa.ini
curl -v -X POST -F "filename=@modelOne.set.New.zip" http://localhost:4040/api/files/file/modelOne.set.New.zip
```

Please notice URL query parameters: `?path=dir%2Fa.ini` which is used to specify `dir/a.ini` path.

PUT create user files folder

PUT create new sub-folder under user files directory on the server.

If user files access enabled in server configuration then user can upload or download files, create or delete folders in user files directory.

This is a beta version and may change in the future.

Method:

```
PUT /api/files/folder/:path
PUT /api/files/folder?path=...
```

Arguments:

```
:path - (required) path to the new sub-directory to be creaded.
```

If sub-directory already exists then method does nothing and return success.

Call examples:

```
curl -v -X PUT http://localhost:4040/api/files/folder/dir
curl -v -X PUT http://localhost:4040/api/files/folder?path=dir%2Fsub-dir
```

Please notice URL query parameters: `?path=dir%2Fsub-dir` which is used to specify `dir/sub-dir` path.

DELETE file or folder from user files

DELETE file or folder from user files directory on the server.

If user files access enabled in server configuration then user can upload or download files, create or delete folders in user files directory.

This is a beta version and may change in the future.

Method:

```
DELETE /api/files/delete/:path
DELETE /api/files/delete?path=....
```

Arguments:

```
:path - (required) path to the new sub-directory to be creaded.
```

If sub-directory not exists then method does nothing and return success.

Call examples:

```
curl -v -X DELETE http://localhost:4040/api/files/delete/dir
curl -v -X DELETE http://localhost:4040/api/files/delete?path=dir%2Fsub-dir
```

Please notice URL query parameters: `?path=dir%2Fsub-dir` which is used to specify `dir/sub-dir` path.

User files may be located above download and upload directory on the server:

```
models/
    home/      -> user personal home directory
        io/       -> user files directory
            download    -> user directory for download files
            upload      -> user directory to upload files
```

Above is a typical server folder structure with user files directory location by default. This method would not delete `download` or `upload` directory if `oms` server is running with downloads or uploads enabled:

```
oms -oms.HomeDir models/home -oms.AllowDownload -oms.AllowUpload
```

DELETE all user files

DELETE all user files from user files directory on the server.

If user files access enabled in server configuration then user can upload or download files, create or delete folders in user files directory.

This is a beta version and may change in the future.

Method:

```
DELETE /api/files/delete-all
```

Call example:

```
curl -v -X DELETE http://localhost:4040/api/files/delete-all
```

User files may be located above download and upload directory on the server:

```
models/
    home/      -> user personal home directory
        io/       -> user files directory
            download    -> user directory for download files
            upload      -> user directory to upload files
```

Above is a typical server folder structure with user files directory location by default. This method would not delete `download` or `upload` directory if `oms` server is running with downloads or uploads enabled:

```
oms -oms.HomeDir models/home -oms.AllowDownload -oms.AllowUpload
```

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 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",
  "DoubleFmt": "%.15g",
  "LoginUrl": "",
  "LogoutUrl": "",
  "AllowUserHome": true,
  "AllowDownload": true,
  "AllowUpload": true,
  "AllowMicrodata": true,
  "IsJobControl": true,
  "IsModelDoc": true,
  "IsDiskUse": true,
  "DiskUse": {
      "DiskScanMs": 11000,
      "Limit": 12884901888,
      "AllLimit": 85899345920
  },
  "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 job service state

GET web-service state.

This method return job service state: model runs queue, active runs and run history.

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
    }
    ],
    "IsDiskUse": true,
    "IsDiskOver": false,
    "DiskScanMs": 11000,
    "Limit": 12884901888,
    "AllLimit": 85899345920
}
```

POST refresh disk space usage info

POST refresh disk space usage info.

This method can be used to refersh disk usage info on the server. Server periodically scan disk storage to GET disk usage state. It is a slow background process and disk usage info may be updated only every 2-3 times per hour. By using this method you can force the server to scan disk usage immediately.

Method:

```
POST /api/service/disk-use/refersh
```

Call example:

```
curl -v -X POST http://localhost:4040/api/service/disk-use/refersh
```

Example:

```
curl -v -X POST http://localhost:4040/api/service/disk-use/refresh

*   Trying [::1]:4040...
* Connected to localhost (::1) port 4040
> POST /api/service/disk-use/refresh HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/8.4.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Location: /api/service/disk-use/refresh/true
< Date: Sat, 10 Feb 2024 08:27:59 GMT
< Content-Length: 0
<
* Connection #0 to host localhost left intact
```

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.

There are 2 method to delete a history records of model run job:

- delete a single job history record by specific model run submission time stamp
- delete all history records of successfuly completed jobs or history of not successful jobs

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.

Methods:

```
DELETE /api/service/job/delete/history/:job

DELETE /api/service/job/delete/history-all/:success
```

Arguments:

```
:job - (required) model run submission time stamp
:success - (required) `true` of `false` boolean flag, use `true` to delete history successfuly completed jobs
```

Call examples:

Delete a history record of job `2022_09_13_23_20_50_995`:

```
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
```

Delete all history records of successfuly completed jobs:

```
curl -v -X DELETE http://localhost:4040/api/service/job/delete/history-all/true

*   Trying [::1]:4040...
* Connected to localhost (::1) port 4040
> DELETE /api/service/job/delete/history-all/true HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/8.4.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Location: /api/service/job/delete/history-all-success/1
< Date: Mon, 06 May 2024 18:46:02 GMT
< Content-Length: 0
```

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 close model database

User can close model database connection, for example, to cleanup database file and reduce storage space. After database connection is closed model would not be available until it is opened again by POST a request to open database file or POST a request to refresh models catalog.

Method:

```
POST /api/admin/model/:model/close
```

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:

```
curl -v -X POST http://localhost:4040/api/admin/model/RiskPaths/close
curl -v -X POST http://localhost:4040/api/admin/model/d90e1e9a49a06d972ecf1d50e684c62b/close
```

For example:

```
curl -v -X POST http://localhost:4040/api/admin/model/d90e1e9a49a06d972ecf1d50e684c62b/close

*   Trying [::1]:4040...
* Connected to localhost (::1) port 4040
> POST /api/admin/model/d90e1e9a49a06d972ecf1d50e684c62b/close HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/8.4.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Location: /api/admin/model/d90e1e9a49a06d972ecf1d50e684c62b/close
< Content-Type: text/plain
< Date: Sat, 09 Mar 2024 04:43:35 GMT
< Content-Length: 0
<
* Connection #0 to host localhost left intact
```

POST a request to delete the model

User can delete all model files to reduce storage space usage. It is a permanent delete, not a "recycle bin" operation and user cannot recover it later.

Method:

```
POST /api/admin/model/:model/delete
```

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:

```
curl -v -X POST http://localhost:4040/api/admin/model/RiskPaths/delete
curl -v -X POST http://localhost:4040/api/admin/model/d976aa2fb999f097468bb2ea098c4daf/delete
```

For example:

```
curl -v -X POST http://localhost:4040/api/admin/model/d976aa2fb999f097468bb2ea098c4daf/delete

> POST /api/admin/model/d976aa2fb999f097468bb2ea098c4daf/delete HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/8.9.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Location: /api/admin/model/d976aa2fb999f097468bb2ea098c4daf/delete
< Content-Type: text/plain
< Date: Fri, 20 Dec 2024 02:00:48 GMT
< Content-Length: 0
<
```

POST a request to open database file

User can open a connection to database file in `models/bin` directory or sub-directories. There is also a method to open all database files in `models/bin` tree: POST a request to refresh models catalog.

Method:

```
POST /api/admin/db-file-open/:path
```

Arguments:

```
:path - (required) path to model database file
```

Path must be relative to `models/bin` folder. If there any subfolders in the path then `/` slashes must be replaced with `*` stars.

Call examples:

```
curl -v -X POST http://localhost:4040/api/admin/db-file-open/RiskPaths.sqlite
curl -v -X POST http://localhost:4040/api/admin/db-file-open/OncoSimX*v3.6.5.2*OncoSimX-allcancers.sqlite
```

For example:

```
curl -v -X POST http://localhost:4040/api/admin/db-file-open/RiskPaths.sqlite

*   Trying [::1]:4040...
* Connected to localhost (::1) port 4040
> POST /api/admin/db-file-open/RiskPaths.sqlite HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/8.4.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Location: /api/admin/db-file-open/RiskPaths.sqlite
< Content-Type: text/plain
< Date: Sat, 09 Mar 2024 05:01:54 GMT
< Content-Length: 0
<
* Connection #0 to host localhost left intact
```

POST a request to cleanup database file

User can reduce database file size by invoking database cleanup script.

Method:

```
POST /api/admin/db-cleanup/:path
POST /api/admin/db-cleanup?path=dir/sub-dir/model.sqlite
POST /api/admin/db-cleanup/:path/name/:name
POST /api/admin/db-cleanup/:path/name/:name/digest/:digest
```

Arguments:

```
:path   - (required) path to model database file
:name   - (optional) model name
:digest - (optional) model digest
```

Path must be relative to `models/bin` folder. If there any sub-folders in the path then `/` slashes must be replaced with `*` stars.

Path, model name and model digest are passed as an arguments to the database cleanup script, for example: Linux script to cleanup database file.

Call examples:

```
curl -v -X POST http://localhost:4040/api/admin/db-cleanup/RiskPaths.sqlite

curl -v -X POST http://localhost:4040/api/admin/db-cleanup?path=OncoSimX/v3.6.5.2/OncoSimX-allcancers.sqlite

curl -v -X POST http://localhost:4040/api/admin/db-cleanup/OncoSimX*v3.6.5.2*OncoSimX-allcancers.sqlite/name/OncoSimX-AllCancers

curl -v -X POST http://localhost:4040/api/admin/db-cleanup/RiskPaths.sqlite/name/RiskPaths/digest/d90e1e9a49a06d972ecf1d50e684c62b
```

Note: You may need to do an URL encoding of your path query parameter: `path=OncoSimX%2Fv3.6.5.2%2FOncoSimX-allcancers.sqlite`

For example:

```
curl -v -X POST http://localhost:4040/api/admin/db-cleanup/RiskPaths.sqlite

*   Trying [::1]:4040...
* Connected to localhost (::1) port 4040
> POST /api/admin/db-cleanup/RiskPaths.sqlite HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/8.4.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Date: Sat, 09 Mar 2024 05:15:49 GMT
< Content-Length: 82
<
{"LogFileName":"db-cleanup.2024_03_09_00_15_49_684.RiskPaths.sqlite.console.txt"}
```

GET the list of database cleanup log(s)

User can reduce database file size by invoking database cleanup script. Console output from cleanup script captured into the log file. This method allow to get the list of database cleanup log files. Use GET database cleanup log file method to get the content of the log file by name.

Method:

```
GET /api/admin/db-cleanup/log-all
```

Example:

```
curl http://localhost:4040/api/admin/db-cleanup/log-all
```


```
[{
    "DbName": "OncoSimX-allcancers.sqlite",
    "LogStamp": "2024_03_05_01_08_24_223",
    "LogFileName": "db-cleanup.2024_03_05_01_08_24_223.OncoSimX-allcancers.sqlite.console.txt"
  }, {
    "DbName": "modelOne.sqlite",
    "LogStamp": "2024_03_05_01_31_56_780",
    "LogFileName": "db-cleanup.2024_03_05_01_31_56_780.modelOne.sqlite.console.txt"
  }, {
    "DbName": "RiskPaths.sqlite",
    "LogStamp": "2024_03_09_00_15_39_343",
    "LogFileName": "db-cleanup.2024_03_09_00_15_39_343.RiskPaths.sqlite.console.txt"
  }, {
    "DbName": "RiskPaths.sqlite",
    "LogStamp": "2024_03_09_00_15_49_684",
    "LogFileName": "db-cleanup.2024_03_09_00_15_49_684.RiskPaths.sqlite.console.txt"
  }
]
```

GET database cleanup log file(s)

User can reduce database file size by invoking database cleanup script. Console output from cleanup script captured into the log file. This method allow to get the content of database cleanup log file by name. Use GET the list of database cleanup log(s) method to obtain log file names.

Method:

```
GET /api/admin/db-cleanup/log/:name
```

Arguments:

```
:name   - (required) database cleanup log file name.
```

Example:

```
curl http://localhost:4040/api/admin/db-cleanup/log/db-cleanup.2024_03_09_00_15_49_684.RiskPaths.sqlite.console.txt
```


```
{
  "DbName": "RiskPaths.sqlite",
  "LogStamp": "2024_03_09_00_29_10_038",
  "LogFileName": "db-cleanup.2024_03_09_00_29_10_038.RiskPaths.sqlite.console.txt",
  "Size": 729,
  "ModTs": 1709962150344,
  "Lines": [
    "2024-03-09 00:29:10.038 etc/db-cleanup_linux.sh models/bin/RiskPaths.sqlite",
    "Model db : RiskPaths.sqlite",
    "Model    : RiskPaths",
    "OM_ROOT  : /home/anatoly/ompp-main",
    "/usr/bin/sqlite3",
    "Source model run status count:",
    "s|1",
    "Update source input scenario: set read-only",
    "Source input scenario count:",
    "1",
    "2024-03-09 00:29:10.259 Model RiskPaths",
    "2024-03-09 00:29:10.296 Model run from 102 RiskPaths_Default to 101",
    "2024-03-09 00:29:10.296   Parameters: 9",
    "2024-03-09 00:29:10.315   Tables: 7",
    "2024-03-09 00:29:10.329 Workset Default from id 101 to 102",
    "2024-03-09 00:29:10.329   Parameters: 9",
    "2024-03-09 00:29:10.341 Done.",
    "Results model run status:",
    "s|1",
    "Results input scenario count:",
    "1",
    "Done: models/bin/RiskPaths.sqlite",
    "2024-03-09 00:29:10.348 Done."
  ]
}
```

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
```

POST a request to pause all model runs queue

POST a request to pause or to resume model runs queue processing for all oms instances

This method is pausing or resuming jobs queue processing for **all** `oms` web-service instances.

For example if there are 3 web-services: `localhost:5050`, `localhost:4040`, `localhost:3030` are sharing the same job queue then:

```
curl -v -X POST http://localhost:4040/api/admin-all/jobs-pause/TRUE
```

would pause jobs from **all** 3 web-services, not only from `localhost:4040`.

This is a beta version and may change in the future.

Method:

```
POST /admin-all/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-all/jobs-pause/true
curl -v -X POST http://localhost:4040/api/admin-all/jobs-pause/1
curl -v -X POST http://localhost:4040/api/admin-all/jobs-pause/0
curl -v -X POST http://localhost:4040/api/admin-all/jobs-pause/false
```

Example:

```
curl -v -X POST http://localhost:4040/api/admin-all/jobs-pause/true

*   Trying 127.0.0.1:4040...
* Connected to localhost (127.0.0.1) port 4040 (#0)
> POST /api/admin-all/jobs-pause/true HTTP/1.1
> Host: localhost:4040
> User-Agent: curl/8.0.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Location: /api/admin-all/jobs-pause/true
< Content-Type: text/plain
< Date: Fri, 04 Aug 2023 00:01:31 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 /shutdown
```

Example:

```
curl -v -X PUT http://localhost:4040/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 /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
```