An API For Microsimulation Models

Graham Stark
Social Work and Social Policy
University of Northumbria
Newcastle, UK
graham.stark@northumbria.ac.uk

***Abstract*—This note describes a simple general Application Programming Interface (API) for controlling microsimulation models.**

***Index Terms*—Microsimulation, APIs**

I. Introduction

An API[1] is a set of standardised rules that allow one piece of software to request and receive information or services from another piece of software. Much of the daily life - online shopping, banking, paying taxes - is built around simple standardised APIs.

This note describes a simple API for interacting with microsimulation models. The initial intended use case for the API is embedding a tax benefit model into an online learning platform; possible other uses include building 'mashups' of simulations from different providers, integrating realistic simulations into games, and running models from inside Content Management Systems (CMSs) such as WordPress.

Standards have been developed for how such APIs should be designed[2] and described[3], and the proposed API tries to adhere to these standards.

II. Online Microsimulation Models

There have been online, publicly available versions of large Microsimulation models since the mid-1990s; the Institute for Fiscal Studies' Be Your Own Chancellor (1995) and Virtual Economy (1999) were early examples. Contemporary examples include the ADRS suite of South African simulations, TriplePC and the University of Essex's UK Mod.

These online models are implemented in different ways. TriplePC has the underlying simulation model and the web interface written in the same programming language (Julia[4]), integrated into a single package. Older systems, and UKMod, have the public facing 'front end' written in a specialist languages like PHP[5] or Java[6], whilst the actual models are developed seperately and invoked as required by the front-end.

Microsimulation models have a number of common characteristics:

- they typically have a large number of inputs, outputs and other controls. It can take dozens of parameters to characterise, for example, an income tax system - tax rates, various allowances, switches for different options and so on;
- healthy models constantly evolve, as they are improved and as the world they try to capture changes. It's rarely a good sign when a model has the same inputs and outputs now as last year;
- they are typically (though not always) resource-intensitve and long running - from a few seconds up to hours or even days. (Even a few seconds is a long time for a typical API service);
- models typically go through a number of distinct phases - sitting in a job queue, initialising, running calculations, generating output, and so on.

Based on our experience since then …

Object - run a model from something like Wordpress - without needing to have the model to hand.

III. Characteristics of Microsimulation Models

Long running
Very different implementations
Phases (queues, running)
Different inputs and outputs
Parameters vs Settings

IV. Features

RESTful (sort of). Reference O'Reilly.
Out of scope: security because …
Learn about exact formats of inputs/outputs
Hacky session management: CORS shit append session_id on each response
Low marginal cost of adding a model (view) to a server
Typically front-ended by Apache/NGNX
Formats: JSON - optionally Markdown/XML/CSV
Describe parameters:
Validate at server end, even if also at client-side.
Swagger.

V. The API

Different for e.g. Julia Scotben, Python Landman so Julia one is:
https://microapi.virtual-worlds.scot
Typical items:

/model/params/set

/model/settings/set

/model/output/fetch/item

Swagger Docs.

A.

B. Problems

buggy!

TABLE I: Others

Benefit	`Code Module`	*Notes*
Minimum Wages	`HouseholdAdjuster.jl`	

References

[1] "API." Accessed: Oct. 23, 2025. [Online]. Available: https://en.wikipedia.org/w/index.php?title=API&oldid=1314301975

[2] M. Masse, *REST API Design Rulebook*. O'Reilly Media, Inc., 2011. Accessed: Oct. 22, 2025. [Online]. Available: https://learning.oreilly.com/library/view/rest-api-design/9781449317904/

[3] "Swagger (software)." Accessed: Oct. 22, 2025. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Swagger_(software)&oldid=1282627605

[4] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Review*, vol. 59, no. 1, pp. 65–98, 2017, doi: 10.1137/141000671.

[5] S. S. Bakken, Z. Suraski, and E. Schmid, *PHP Manual: Volume 1*. iUniverse, Incorporated, 2000.

[6] K. Arnold, J. Gosling, and D. Holmes, *The Java programming language*. Addison Wesley Professional, 2005.