Jul 12th, 2:50 PM - 3:10 PM

Environmental Models as a Service: Enabling Interoperability through RESTful Endpoints and API Documentation

Jonathan Flaishans  
Oak Ridge Institute for Science and Education, flaishans.jonathan@epa.gov

Mike Galvin  
U.S. Environmental Protection Agency, galvin.mike@epa.gov

Amber Ignatius  
Oak Ridge Institute for Science and Education, ignatius.amber@epa.gov

Carmen Kuan  
Oak Ridge Institute for Science and Education, kuan.carment@epa.gov

Gerard F. Laniak  
U.S. Environmental Protection Agency, laniak.gerry@epa.gov

Follow this and additional works at: https://scholarsarchive.byu.edu/iemssconference

Part of the Civil Engineering Commons, Data Storage Systems Commons, Environmental Engineering Commons, Hydraulic Engineering Commons, and the Other Civil and Environmental Engineering Commons

Flaishans, Jonathan; Galvin, Mike; Ignatius, Amber; Kuan, Carmen; Laniak, Gerard F.; Wolfe, Kurt; and Purucker, Tom, "Environmental Models as a Service: Enabling Interoperability through RESTful Endpoints and API Documentation" (2016). International Congress on Environmental Modelling and Software. 18.  
https://scholarsarchive.byu.edu/iemssconference/2016/Stream-A/18

This Event is brought to you for free and open access by the Civil and Environmental Engineering at BYU ScholarsArchive. It has been accepted for inclusion in International Congress on Environmental Modelling and Software by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.
Environmental Models as a Service: Enabling Interoperability through RESTful Endpoints and API Documentation

Jonathan Flaishans\textsuperscript{a}, Mike Galvin\textsuperscript{b}, Amber Ignatius\textsuperscript{a}, Carmen Kuan\textsuperscript{a}, Gerard F. Laniak\textsuperscript{b}, Kurt Wolfe\textsuperscript{b}, Tom Purucker\textsuperscript{b}

\textsuperscript{a}Oak Ridge Institute for Science and Education (Flaishans.Jonathan@epa.gov, Ignatius.Amber@epa.gov, Kuan.Carmen@epa.gov)

\textsuperscript{b}U.S. Environmental Protection Agency (Galvin.Mike@epa.gov, Laniak.Gerry@epa.gov, Wolfe.Kurt@epa.gov, Purucker.Tom@epa.gov)

Abstract: Achieving interoperability in environmental modeling has evolved as software technology has progressed. The recent rise of cloud computing and proliferation of web services initiated a new stage for creating interoperable systems. Scientific programmers increasingly take advantage of streamlined deployment processes and affordable cloud access to move algorithms and data to the web for discoverability and consumption. In these deployments, environmental models can become available to end users through RESTful web services and consistent application program interfaces (APIs) that consume, manipulate, and store modeling data. RESTful modeling APIs also promote discoverability and guide usability through self-documentation. Embracing the RESTful paradigm allows models to be accessible via a web standard, and the resulting endpoints are platform- and implementation-agnostic while simultaneously presenting significant computational capabilities for spatial and temporal scaling. RESTful APIs present data in a simple verb-noun web request interface: the verb dictates how a resource is consumed using HTTP methods (e.g., GET, POST, and PUT) and the noun represents the URL reference of the resource on which the verb will act. The RESTful API can self-document in both the HTTP response and an interactive web page using the Open API standard. This lets models function as an interoperable service that promotes sharing, documentation, and discoverability. Here, we discuss the process of making a set of science models used by the EPA for pesticide registration available as RESTful endpoints and provide an example API for environmental models, using U.S. Environmental Protection Agency pesticide registration models.

Keywords: Application Programming Interface; RESTful; web services; model as a service

1 INTRODUCTION

Interoperability of environmental modeling frameworks has evolved from client side coupling to the use of increasingly distributed systems and standardization of data transfer protocols. Continued evolution of computational systems technologies alters the discovery, access, and integration approaches available to solve interoperable environmental modeling problems (Ignatius et al., 2016, Laniak et al., 2013). A common distributed approach is construction of a RESTful application program interface (API) that complies with a REST (REpresentational State Transfer) software architectural style (Fielding et al., 2000). RESTful implementations powering a majority of public services on the modern internet while leveraging a data as a service model (Richardson and Ruby, 2007). RESTful APIs are used to document scientific data as a service (e.g., Ignatius et al. 2016, Parmar et al. 2016, Wolfe et al. 2016), but can also be used to create science models as a service. This approach provides a platform where models are accessible to users over the internet while being platform-agnostic. In addition, self-documentation of a RESTful API promotes discoverability of its services through navigable or interactive interfaces, and the simplification and
standardization of RESTful architecture allows practical integration across systems. Combined with the rise of cloud computing and platforms as a service (PaaS) these approaches can simplify the process of developing and deploying environmental models and data as a service.

The übertool is a suite of web-based ecological risk assessment models supported by the United States Environmental Protection Agency (USEPA) that provides a cross-platform implementation for users. Its web-based approach provides users with a common interface to models developed in differing formats ranging from simple spreadsheet calculators to platform-dependent compiled executables. The suite combines features of models into a unified, web-based experience by utilizing a template-based, modular design. The modular structure allows coupling of multiple models with minimal code and the use of native source code with Python wrappers and RESTful servers (Flaishans et al., 2014, Hong et al., 2014). In addition to the frontend (graphical interface) implementation, expressing the übertool models as an API can increase discovery, access, and integration of pesticide risk assessment models used by the USEPA for other model developers. The übertool API implementation demonstrates how a governmental agency can provide environmental modeling as a service through a RESTful API backend, a web application frontend, and interactive API documentation. The RESTful API backend hosts the environmental models as a service, which are accessible using the JavaScript Object Notation (JSON) communication standard. The frontend web site consumes the RESTful API backend and offers access to pesticide models through an intuitive user interface. Interactive API documentation provides users with technical information in order to use backend RESTful APIs in other applications. The übertool API can be leveraged as an interoperable system built on a standard RESTful platform to provide environmental modeling to both public and private users.

2 METHODS

2.1 RESTful APIs

To promote discoverability of web services in an integrated environmental modeling API, a RESTful interface can be implemented to expose the models to users over the Internet. A RESTful API applies the engineering principle of generality to simplify the interface and increase visibility of available interactions between the service and the user (Fielding, 2002). This simplicity is accomplished by emphasizing a uniform communication interface between components. RESTful APIs represent data (in our case, environmental models) as “resources” and, more generally, as collections of those resources. The components are referred to as REST data elements. Each collection of resources is located by an identifier which is a web address or uniform resource locator (URL). A resource is consumed by a Hypertext Transfer Protocol (HTTP) request sent from the client, with the requested data returned in a HTTP response. Each HTTP response has metadata describing its media type, last-modified time, source link, etc. in a HEADER element, and the return data consumed in a BODY element by the user.

RESTful APIs were designed to be general, uniform solutions to a multitude of problems related to creation and consumption of data over the Internet, not explicitly for scientific modeling or environmental data representation. Instead, RESTful APIs are built upon basic components of modern communication between clients and servers over the internet using HTTP (Fielding et al., 1999). HTTP includes a set of methods that act upon resources; the most common are represented by the verbs POST, GET, PUT, and DELETE. For a RESTful API, most users conceive of the use cases of these methods by mapping them to basic database CREATE, READ, UPDATE, and DELETE operations (CRUD). However, the HTTP verbs offer more flexibility than their CRUD counterparts because of generality in definition. For example, while POST and GET requests can both return data corresponding to a database READ operation to a user, there is a difference in how the HTTP request data is sent to the server. POST sends data in the request body as form data or a string format (usually JSON); GET sends data as part of a query string in the URL as web servers generally ignore the BODY of GET requests. A major implication of these differences is security. A request BODY is encrypted when sent over secure socket layer (SSL) protocols using HTTPS, protecting the user’s data; therefore, POST requests are the preferred method for sensitive data transfer.
RESTful APIs represent data as resources, and available HTTP methods dictate the actions on those resources. This creates a noun-verb interaction between resources and HTTP methods, defining RESTful services. The verbs are limited and correspond to available methods. Such limited availability in verbs, or actions, for each resource is in stark contrast and reaction to the Simple Object Access Protocol (SOAP), which was the dominant protocol for information exchange over the internet prior to REST. SOAP allowed unlimited, custom-defined actions for each resource, which led to confusing, non-standard actions that required increasing amounts of documentation (Richardson and Ruby, 2007). Limiting actions to predefined HTTP methods standardizes RESTful services and provides a common lexicon. Nouns are reserved for the data resources provided by the web service, referenced by their URLs. An example of the noun-verb structure of a RESTful API is shown in Table 1.

The übertool API follows these conventions and offers environmental models and associated data as resources. The API guides users on how to consume web services through its design and, more importantly, through its documentation. Documentation is common practice for RESTful APIs, but documenting scientific models generally requires more information than web services which offer relatively straightforward database operations. Environmental models require documentation to define input and output parameters, their units and valid ranges, as well as explain how algorithms are implemented. This extra burden of scientific information requires environmental modeling APIs to have more comprehensive technical and scientific documentation.

<table>
<thead>
<tr>
<th>Table 1. Model API HTTP Request and Response Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Request Object</strong></td>
</tr>
<tr>
<td><strong>HTTP Method</strong></td>
</tr>
<tr>
<td>GET</td>
</tr>
<tr>
<td>GET</td>
</tr>
<tr>
<td>GET</td>
</tr>
<tr>
<td>POST</td>
</tr>
</tbody>
</table>

¹All Resource URLs are prepended with “https://qed.epa.gov/rest/ubertool/{model_name}”

2.2 API Documentation for Environmental Models as a Service

2.2.1 Open API Interactive Documentation

The übertool API documentation consists of an Open API interactive RESTful API and scientific model metadata. RESTful API documentation is a common feature of web services, documenting how available HTTP methods interact with data (resources) contained in a web service. Standardized frameworks are available to generate API documentation and simplify documentation generation and deployment. We utilize an open source solution (Open API) to interactively document available web services. The RESTful API documentation (available at https://qed.epa.gov/api) is a graphical interface to explore the data and models provided for the übertool models, as shown in Fig. 1.

RESTful API documentation provides an interactive list of all available endpoints for each model. A model is a collection resource which allows GET and POST operations on a set of specific resources (Table 1). The API provides inputs required to execute a model by performing a GET request at “/rest/ubertool/<model_name>/inputs”, which returns a JSON string representing the schema required
to run the model. The output schema from a successful model run can be obtained by performing a GET request at "/rest/ubertool/<model_name>/outputs", which also returns as a JSON string. A GET request to the collection resource itself, e.g. "/rest/ubertool/<model_name>“, returns a JSON string documenting all the endpoints and further available actions. This practice follows the Hypermedia as the Engine of Application State (HATEOAS) constraint of RESTful APIs (Fielding, 2000). HATEOAS allows a RESTful API to be navigated by explicitly stating all the actions and resources the current endpoint can reach. This is accomplished by embedding hypermedia references to each reachable state in the endpoint’s response (Fig. 2). The interactive Open API documentation page allows users to test and demo web services from the web browser. Each endpoint has a “Try it out!” button that shows responses from live requests. Model runs can be executed via POST requests to give developers a feel for the available web services.

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>agdrift</td>
<td>Agdrift Model</td>
</tr>
<tr>
<td>GET /rest/ubertool/agdrift/</td>
<td>Returns Agdrift JSON schema</td>
</tr>
<tr>
<td>POST /rest/ubertool/agdrift/{jobId}</td>
<td>Agdrift Model</td>
</tr>
<tr>
<td>earthworm</td>
<td>Earthworm Model</td>
</tr>
<tr>
<td>GET /rest/ubertool/earthworm/</td>
<td>Returns Earthworm JSON schema</td>
</tr>
<tr>
<td>POST /rest/ubertool/earthworm/{jobId}</td>
<td>Earthworm Model</td>
</tr>
<tr>
<td>iew</td>
<td>Iec Model</td>
</tr>
<tr>
<td>iew</td>
<td>Rice Model</td>
</tr>
<tr>
<td>sip</td>
<td>Sip Model</td>
</tr>
<tr>
<td>stir</td>
<td>Stir Model</td>
</tr>
<tr>
<td>terrplant</td>
<td>Terrplant Model</td>
</tr>
</tbody>
</table>

![übertool API Documentation](image)

**Fig. 1. Interactive Open API übertool RESTful API documentation.**

### 2.2.2 API Self-documentation

We follow API self-documentation practices described by the HATEOAS constraint of RESTful services. Self-documentation adds discoverability to resources contained in the API, as well as a human- and computer-navigable interface to RESTful endpoints. Self-documentation of endpoints is located at the root of all available RESTful endpoints: “https://qed.epa.gov/rest.” From this location, all possible endpoints branch out to form a structure similar to a file transfer protocol (FTP) or file explorer client, aiding in discovery and accessibility of the API. Included in each response from a GET request for any particular endpoint is a JSON string containing metadata -- namely, a hypermedia reference (href) to all accessible branches from that endpoint. Semantics of the HATEOAS implementation allow software consuming the web service to always refer to available actions from the current state. For example, a GET request to the model TerrPlant
at “https://qed.epa.gov/rest/ubertool/terrplant” returns a JSON string response containing hrefs (Fig. 2). The “link” object contains all API endpoints relative to that resource collection. In the TerrPlant model, the response informs the user of URL endpoints to acquire model inputs, model outputs, the “execute” endpoint, a hypertext link to the TerrPlant web page, and location of the endpoint that returned the response.

```json
{
  "metaInfo": {
    "model": "terrplant",
    "collection": "ubertool",
    "url": {
      "ref": "self",
      "href": "https://qed.epa.gov/rest/terrplant"
    },
    "modelVersion": "1.2.2",
    "timestamp": "2016-06-07T14:37:05,279092093-0400",
    "description": "TerrPLANT provides screening level estimates of exposure to terrestrial plants from single pesticide applications through runoff or drift"
  },
  "links": [
    {
      "ref": "inputs",
      "type": "application/json",
      "href": "https://qed.epa.gov/rest/terrplant/inputs"
    },
    {
      "ref": "outputs",
      "type": "application/json",
      "href": "https://qed.epa.gov/rest/terrplant/outputs"
    },
    {
      "ref": "run",
      "type": "application/json",
      "href": "https://qed.epa.gov/ubertool/terrplant/run",
      "method": "POST"
    },
    {
      "ref": "html",
      "type": "text/html",
      "href": "https://qed.epa.gov/ubertool/terrplant"
    }
  ]
}
```

**Fig. 2.** Example JSON response from RESTful model endpoint “https://qed.epa.gov/rest/ubertool/terrplant”

### 2.2.3 Scientific Documentation

In addition to the technical documentation of the RESTful API, supporting scientific documentation for each model is provided. This includes resources such as user guides, manuals, and peer-reviewed publications. The web site contains links to these materials as well as detailed explanations of model algorithms. Web site users can navigate to “References” and “Algorithms” for each model to view relevant scientific information. The RESTful web service contains hypertext links to model web pages for scientific documentation.
The complex nature of environmental, process-based models does not lend itself to user-facing documentation when presented in a raw string format such as JSON. Instead of trying to document scientific processes and algorithms using a machine-readable technology (the HTTP response object), the übertool API gives users a hypertext link reference to comprehensive model documentation hosted on the frontend website. This allows existing documentation and publications to be reused and gives more context to reference materials. A comprehensive collection of all reference materials, each with its own hypermedia reference included in the HTTP response, is an option for future improvements to the RESTful API.

2.3 übertool Models as a Service

The RESTful API resources presented here are a collection of USEPA pesticide registration models and data known as the übertool. Before being implemented as API endpoints and web applications, übertool models were in software formats ranging from spreadsheet calculators to FORTRAN 95/2003 executables. They have been recoded as needed to allow for migration to a cloud implementation. Models were either ported to Python code, or Python wrapper code was written to expose the legacy code and then served through a backend Python Flask web server (Hong et al., 2014). Web application frontends were created for users of legacy model implementations to have a familiar user interface (Flaishans et al., 2014). The frontend is also the resource hub for documentation and references for the model.

Aside from simple spreadsheet calculators, the übertool also hosts large-scale spatial models that consume and produce large amounts of data. Cloud implementation of the RESTful server allows model optimization with parallelization, and horizontal-scaling with load-balancing. The Spatial Aquatic Model (SAM) is a data-intensive regional-scale pesticide exposure to waterbodies model that is part of the übertool. SAM was originally a FORTRAN 95-based model converted to a dynamic-link library (DLL) / shared library (.so) that was accessible by the RESTful API. Runs of the regional-scale model were long, especially for a web service, typically lasting upwards of an hour. Through parallelization and optimization, typical SAM runs now take 30-60 seconds in the cloud environment (Flaishans et al., 2016), a 98-99% speed increase that highlights the power of cloud computing and benefits of hosting environmental models as RESTful APIs.

3 DISCUSSION

Hosting models as a service provides additional benefits for application users and developers beyond traditional scientific desktop applications. Users gain a consistent interface, updated implementation and an easily discoverable location for multiple pesticide registration models used by the USEPA. This combination increases discoverability and transparency of the science and process of pesticide risk assessment while allowing for more substantial data inputs and workloads than are available within a typical desktop environment. With the RESTful API, developers gain a unified approach to serving models and data based on an industry-proven standard emphasizing simplicity and self-documentation. RESTful APIs for environmental models create an interoperable interface to share scientific algorithms and process modeling with the larger environmental developer community. Examples of interoperable use of an environmental RESTful modeling API include web application frontends such as the one at https://qed.epa.gov/ubertool, and integration into existing client-side, server-side or web-based modeling frameworks.

RESTful APIs do not force or supply environmental modeling specific standards because their services standardize at a more basic level of internet communication. This gives researchers and model developers more freedom to customize services for problems they are solving. Documentation becomes essential with this loose protocol approach. Environmental RESTful modeling APIs such as the übertool should embrace this and document at technical and scientific levels to enable access for users and developers.

When being considered for an integrated modeling system, RESTful APIs have downsides that must be weighed against their benefits of increased discovery, access, and integration. Long-term support and costs of operating the RESTful service can be problematic in grant-funded situations, although the price of cloud hosting drops steadily year after year and platforms as a service provide free tiers of cloud hosting. A trade-
off also exists between providing models as a service, with its potential speed bottleneck by communicating in raw strings over the Internet, and interoperability between services relying on a web standard. A more tightly coupled system can be optimized for maximum speed of data communication at the cost of interoperability between it and others. RESTful APIs will not be the most speed-efficient solution of a process-driven system when individual processes require their own HTTP request and response cycles. Providing models as a service can overcome this perceived downside by allowing a set of user inputs to dictate process coupling server-side without the need for HTTP communication between client and server for each process, although this approach inhibits the system from operating interoperable with other services. Additionally, a model relying on multiple decision trees with variable inputs, such as one requiring a complex user interface, is challenging to represent clearly and efficiently through a JSON-based RESTful API.

This implementation of a RESTful API has elevated a collection of environmental models with disparate technologies to a unified one with a service framework. Offering interoperability that utilizes standard HTTP communication protocols and documentation, this implementation represents modernization of environmental modeling and data for longstanding regulatory processes used by the USEPA. In addition to enhancing interoperability of the key model components, the resulting übertool platform implementation makes the science behind the regulatory process more scalable, discoverable, extendable and transparent.

REFERENCES


