

UWS through OpenAPI

Joshua Fraustro May 22nd, 2024



STSCI | SPACE TELESCOPE SCIENCE INSTITUTE

EXPANDING THE FRONTIERS OF SPACE ASTRONOMY





What would a UWS 1.1 OpenAPI specification look like? Why UWS?

- A sufficiently complicated example of a web service pattern that is already very **RESTful in its design.**
- No part of it requires describing / modeling data formats. (VOTables)

Demonstrate:

STSCI | SPACE TELESCOPE SCIENCE INSTITUTE

- Paths, operations, parameters, and protocol models could be adequately represented.
- Version changes, iterative and large updates, were easy to create and work against. We could take advantage of modern tooling that uses OpenAPI standards.

UWS through OpenAPI, Joshua Fraustro, 2024

Problems? What problems?

previous Interops.

Some are small and simple to fix, some are definitely breaking

- Case-insensitive query parameters
- 303 HTTP status codes for successful operations (creating Jobs)
- POST operations for updating Job parameters
- Difficult to describe nuances of the XML schema
- Empty response bodies for certain Job parameters
- Unclear which parameters can be updated with POSTS to their endpoint.

See: P3T Sydney 2024 & DAL Bologna 2023

UWS through OpenAPI, Joshua Fraustro, 2024

STSCI | SPACE TELESCOPE SCIENCE INSTITUTE

Some of the problems & HTTP/REST issues in UWS have been pointed out in





I presented a version of UWS in OpenAPI in Sydney & virtually. This version had the OpenAPI doc + all of the "fixes". Generally:

- People liked the idea of adding this kind of documentation.
- Felt it added an extra option for client & service developers.

But also concerns:

STSCI | SPACE TELESCOPE SCIENCE INSTITUTE

Too much, too fast, breaking changes, interoperability concerns

How do we bridge the gap from here, to there?

UWS through OpenAPI, Joshua Fraustro, 2024

Describing UWS in 3 steps

As an exercise, create 3 versions of the spec, showing we can iterate:

" As-Is "

 Describe the current UWS 1.1 standard as closely as possible

• Only make changes that are not otherwise possible to avoid.

STSCI | SPACE TELESCOPE SCIENCE INSTITUTE

UWS through OpenAPI, Joshua Fraustro, 2024

"Refinement"

- Solve a few problems with small changes
 - Non-breaking / as easy for client developers as possible

"What-If?"

- Solve the highlighted problems, knowing things will break
- Replace XSD definitions with **OpenAPI** schema models





Version 1:

STSCI | SPACE TELESCOPE SCIENCE INSTITUTE

A mostly straight-forward description of UWS 1.1 into an OpenAPI specification. **Positives:**

- Fairly simple to do, UWS is already very RESTful in its design.
- Notes:
 - OpenAPI 3.0 has XML support, but too basic for the UWS schema.
 - Case-insensitive query parameters are not describable.
 - Redocly linting immediately noticed HTTP status code problem.

UWS through OpenAPI, Joshua Fraustro, 2024

Paths, operations easy to document — no conflict with OpenAPI specs.

Avoid trying to model the XML request / responses, only point to current XSD.





	paths: /	{job-id}:
	/:	parameters:…
	/{job-id}:	get:
	parameters:	Scan Try it Audit
	get:	operationId: getJobSu
-	post:	summary: 'Returns the
	delete:	parameters:
	/{job-id}/phase:	responses: '200':
	parameters:	description: Suco
	get:	content:
	post:	application/xm
	/{job-id}/executiondur	schema
	parameters:	\$ref: ' <u>#/co</u>
	get:	'403':
	nost	<pre>\$ref: '#/componer</pre>
1000	"As-Is" OpenAPI Document	404':
		sref: <u>#/componer</u>
	Examples here of: A Beth peremotors and operations	post:
	Path parameters and operations Page opumaration	
	Response enumeration Response enumeration	
	Request & response models Gitbub Link:	
	Github Link:	
	https://github.com/jwfraustro/PTTT/tree	<u>Juws-Dasic</u>

Carlos and the second



Summary e job summary'

cess

l:

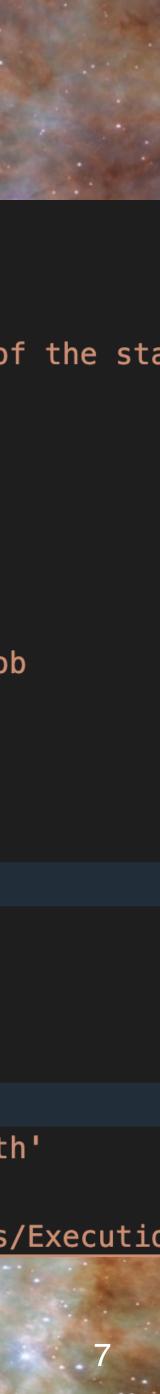
components/schemas/Job'

ents/responses/Forbidden'

nts/responses/JobNotFound



JobSummary: type: object description: The complete representation of the sta title: jobSummary required: [jobId] properties: jobId: type: string description: The identifier for the job example: 'HSC_XYZ_123' runId: type: string maxItems: 1 description: | --example: 'JWST-1234' ownerId: type: string nullable: true description: | --example: 'Noirlab/John.Smith' phase: \$ref: '#/components/schemas/Execution





STScI | SPACE TELESCOPE SCIENCE INSTITUTE

• Minimal to implement for services, clients. An example of how it's easy to see changes in the OpenAPI document.

Change	Example	Breaking?
All query parameters shall be lowercase or camelCase (for multi-word params)	PHASE -> phase OtherParam -> otherParam	No. DALI allows arbitrary casing.
HTTP status codes for GET's to empty parameters should indicate as such	GET /quote 204, No-Content or 200, null	Yes, but a simple change.
303 Redirects (for POST's, etc.) changed to their appropriate status codes	POST /jobs should return 200 OK	Yes, but a simple change.

UWS through OpenAPI, Joshua Fraustro, 2024

Version 2: Small changes to fix design issues without greatly changing the spec.



	paths:			
•	/:	summary: Returns the list of UWS jobs		
	get:	parameters:	ALL STREET, SALES	
		<pre>- name: PHASE</pre> /{job-id}/destruction:		
	Scan Try it Audit	– name: phase You, 8 m		
	<pre>operationId: getJobList</pre>	in: query - \$ref: '#/components	s/parameters/job-id'	
	summary: Returns the list of UWS jobs	description: 'Execution r get:		
	parameters:	schema: operationId: getJobDe	estruction	
	- name: phase…		e job destruction time'	
	- name: after…	- name: AFTER responses:		
	– name: last…	- name: after '200':		
	responses:	in: query description: Succ	cess	
-		description: 'Return jobs content:		
	post:	schema: text/plain:		
	Scan Try it Audit	type: string schema:		
	<pre>operationId: postCreateJob</pre>	format: date-time type: strir	ng	
	summary: 'Submits a job' -	– name: LAST format: dat	.	
	requestBody:	- name: last		
	responses	in: query description: 'No	destruction time set'	
	'200':	description: Return only		
	·////	schellia:		
		enum:		
	"Refinement" OpenAPI Document			
Į.,				
	 New OpenAPI document on the left 	- "SUSPEND"		
	 Git diffs between the two versions are easy to see 	- "ARCHIVE"		
	and understand.	'303':		
		200':		
	Github Link:	description: "Success"		
	https://github.com/jwfraustro/PTTT/tree/uws-improved	<pre>\$ref: '#/components/responses/JobSummaryRedirect'</pre>		
		<pre>\$ref: '#/components/responses/JobSummaryResp</pre>		

and the second second





Version X: The version I presented in Sydney. Changes:

- All of the previous changes
- Request / response job messages are fully JSONSchema describable.
 - Means we get more varied native encoding formats.
- Job creation by POST'ing the document no HTTP/form-encoded

Is it breaking?

• Well, yes.

STSCI | SPACE TELESCOPE SCIENCE INSTITUTE

Working on a prototype implementation at MAST w/ FastAPI client libraries.

post: tags: [UWS] requestBody: required: true content: schema:

"What-If" OpenAPI Document

- Removal of form-url encoded.
- Simple to represent change, just point the

request body at the 'Parameters' object

Github Link:

https://github.com/jwfraustro/PTTT/tree/UWS-MAST

and the second second

```
operationId: postCreateJob
summary: 'Submits a job'
 description: 'Job parameters'
    application/x-www-form-urlencoded:
    application/json:
        type: object
        properties:
          # Examples for TAP implementation
          QUERY:
            type: string
            description: 'The query to be performed'
            example: 'SELECT * FROM TAP_SCHEMA.tables'
          LANG:
            type: string
            description: 'The language in which the query should be performed'
            example: 'ADQL'
        additionalProperties: true
        $ref: '#/components/schemas/Parameters'
```







Demonstrations here are: Not about how easy the changes are, but how easy they are to document.

Show that:

STSCI | SPACE TELESCOPE SCIENCE INSTITUTE

- We can take iterative steps towards what was proposed.
- Iteration is clearly shown through these documents.
- We have a version that can be deployed "tomorrow".
- We understand what might break at each step.

UWS through OpenAPI, Joshua Fraustro, 2024



What's next?

- TAP 1.2 will be implementing an OpenAPI spec that needs UWS.
- Take a look at the "As-Is" version of the OpenAPI spec.
 - No functional changes in the standard.
 - Needs review from more eyes (than mine!)
- Think about how we would integrate OpenAPI docs with our current document publishing pipelines.
- Keep pushing forward with JSON-compatible implementation / libraries to understand what the future looks like.



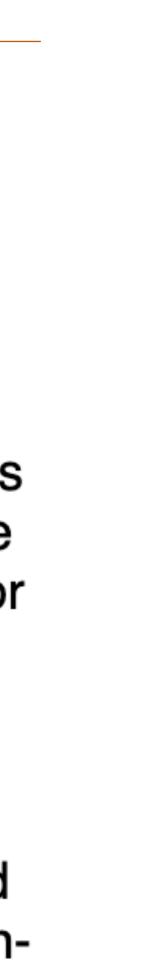
IVOA-style standards document for the classic PetStore API example:

2.2.3.1. Getting the Pet List

The list of pets available in the Pet Store may be retrieved by sending a GET request to the endpoint /pets. In this case, the query parameters that may be included in the request are LIMIT, which restricts the number of pet items returned, and STATUS, which allows the client to specify a filter based on the current availability of the pets. The status parameter accepts the values "AVAILABLE", "PENDING", or "SOLD". The response to this request will contain a JSON array of pet objects, each representing a distinct pet record in the system.

Upon successful retrieval, the response code will be 200 "OK", and the response body will contain a JSON representation of the pets matching the query parameters, if any. The server may also respond with a 400 "Bad Request" status code if the request parameters are invalid (for example, if limit is non-numeric or negative).

STSCI SCIENCE INSTITUTE UWS through OpenAPI, Joshua Fraustro, 2024



Bonus: PetStore IVOA Spec

Can you diff it?

- 1 2.2.3.1. Getting the Pet List
- 2
- 3 The list of pets available in the Pet Store may be retrieved by sending a GET request to the endpoint /pets. In this case, the query parameters that may be included in the request are LIMIT, which restricts the number of pet items returned, and STATUS, which allows the client to specify a filter based on the current availability of the pets. The status parameter accepts the values "AVAILABLE", "PENDING", or "SOLD". The response to this request will contain a JSON array of pet objects, each representing a distinct pet record in the system.
- 4
- 5 Upon successful retrieval, the response code will be 200 "OK", and the response body will contain a JSON representation of the pets matching the query parameters, if any. The server may also respond with a 400 "Bad Request" status code if the request parameters are invalid (for example, if limit is non-numeric or negative). UWS through OpenAPI, Joshua Fraustro, 2024 ST SCIENCE INSTITUTE

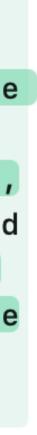
1 2.2.3.1. Getting the Pet List

2

3 A client may access the list of pets in the Pet Store by initiating a GET request to /pets. Clients can include optional query parameters in the request to influence the returned data. These parameters include QUANTITY, which specifies the desired count of pet items in the response, and STATE, which allows clients to specify a filter based on the current state of the pets, accepting values such as "ACTIVE", "RESERVED", or "SOLD". The server's response will include a JSON-formatted array of objects, each representing a pet with various attributes.

4

5 On successful execution of the request, the server will return a status code of 200 "OK", with a JSON body representing the list of pets that meet the specified criteria. The server may return a 400 "Bad Request" if it detects invalid parameter values, such as a non-numeric quantity or unrecognized state value.







Bonus: PetStore IVOA Spec

<pre>/pets: get: tags: Can you diff the OpenAPI spec? - pet summary: List all pets description: Returns all pets from the store operationId: listPets parameters:</pre>
- name: limit
in: query description: How many pets to return at one time (max 100) required: false schema: type: integer format: int32
– name: <mark>status</mark>
<pre>in: query description: Filter pets by status required: false schema: type: array items: type: string enum:</pre>
<pre>- available - pending</pre>
- sold responses:

responses:

STSCI | SPACE TELESCOPE SCIENCE INSTITUTE UWS through OpenAPI, Joshua Fraustro, 2024

```
/pets:
 1
 2
       get:
 3
         tags:
           - pet
 4
         summary: List all pets
 5
         description: Returns all pets from the store
 6
         operationId: listPets
 7
         parameters:
 8
           - name: quantity
 9
10
             in: query
11
             description: How many pets to return at one time (max 100)
             required: false
12
13
             schema:
14
               type: integer
               format: int32
15
16
           - name: state
17
             in: query
             description: Filter pets by status
18
             required: false
19
20
             schema:
               type: array
21
22
               items:
23
                 type: string
24
                 enum:
25
                   - active
26

    reserved

                   - sold
27
28
         responses:
```

