CDS UWS Library

Thomas Boch, on behalf of

- *Grégory Mantelet*
- *Brice Gassmann*

UWS

Universal Worker Service pattern

- **What is it ?**

A pattern for a Web-Service which has to manage asynchronous jobs.

- **How does it work ?**

1 URL => 1 Action

- **What are the possible actions ?**

Create a job, Set job parameters, Execute a job, Get job results, Get list of jobs, etc...

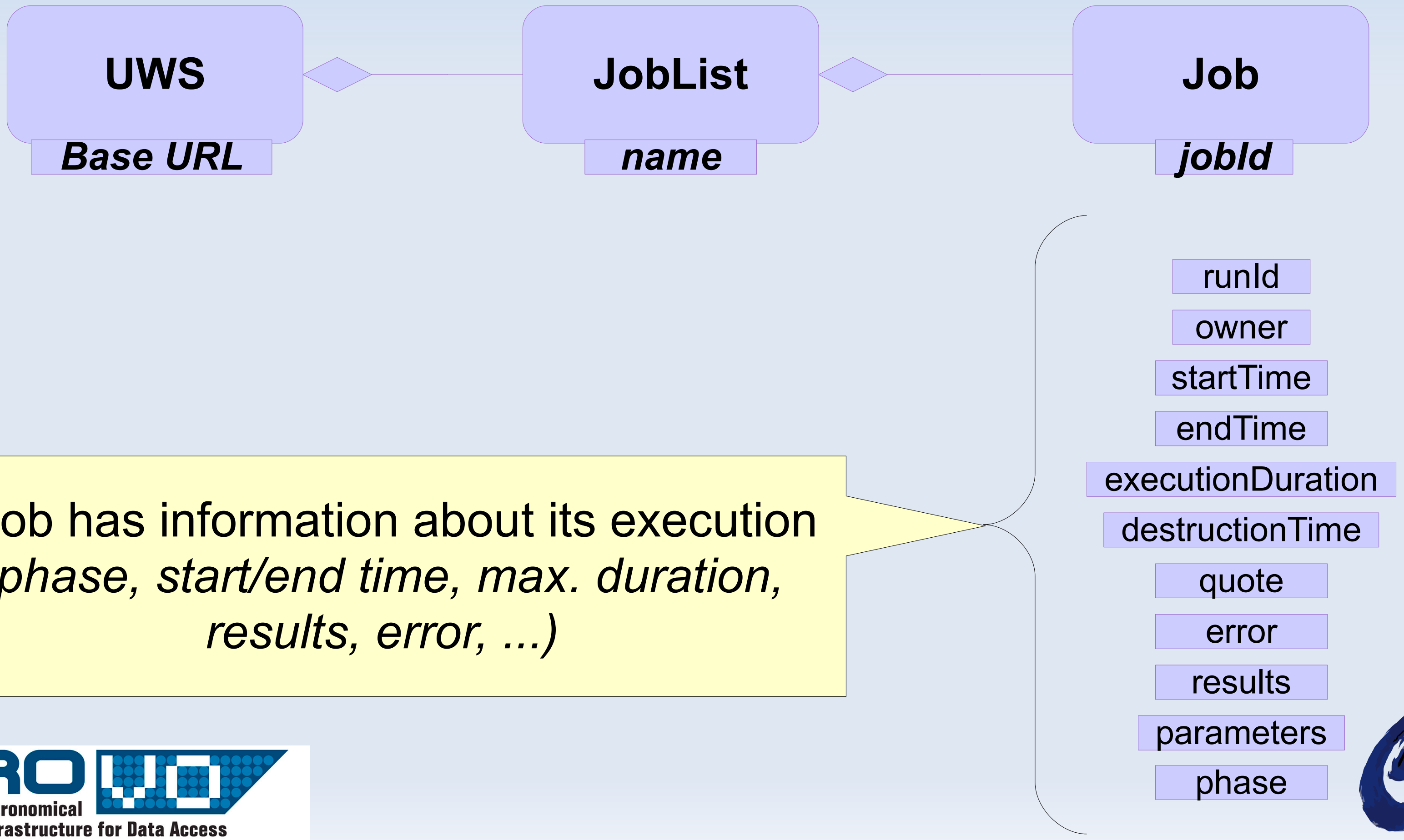
EURO VO AIDA Astronomical Infrastructure for Data Access

CENTRE DE DONNÉES ASTRONOMIQUES DE STRASBOURG

Quick description

UWS as a Tree

- A UWS is structured as a tree, in which a job is a leaf.
- Each node is considered as a web resource...

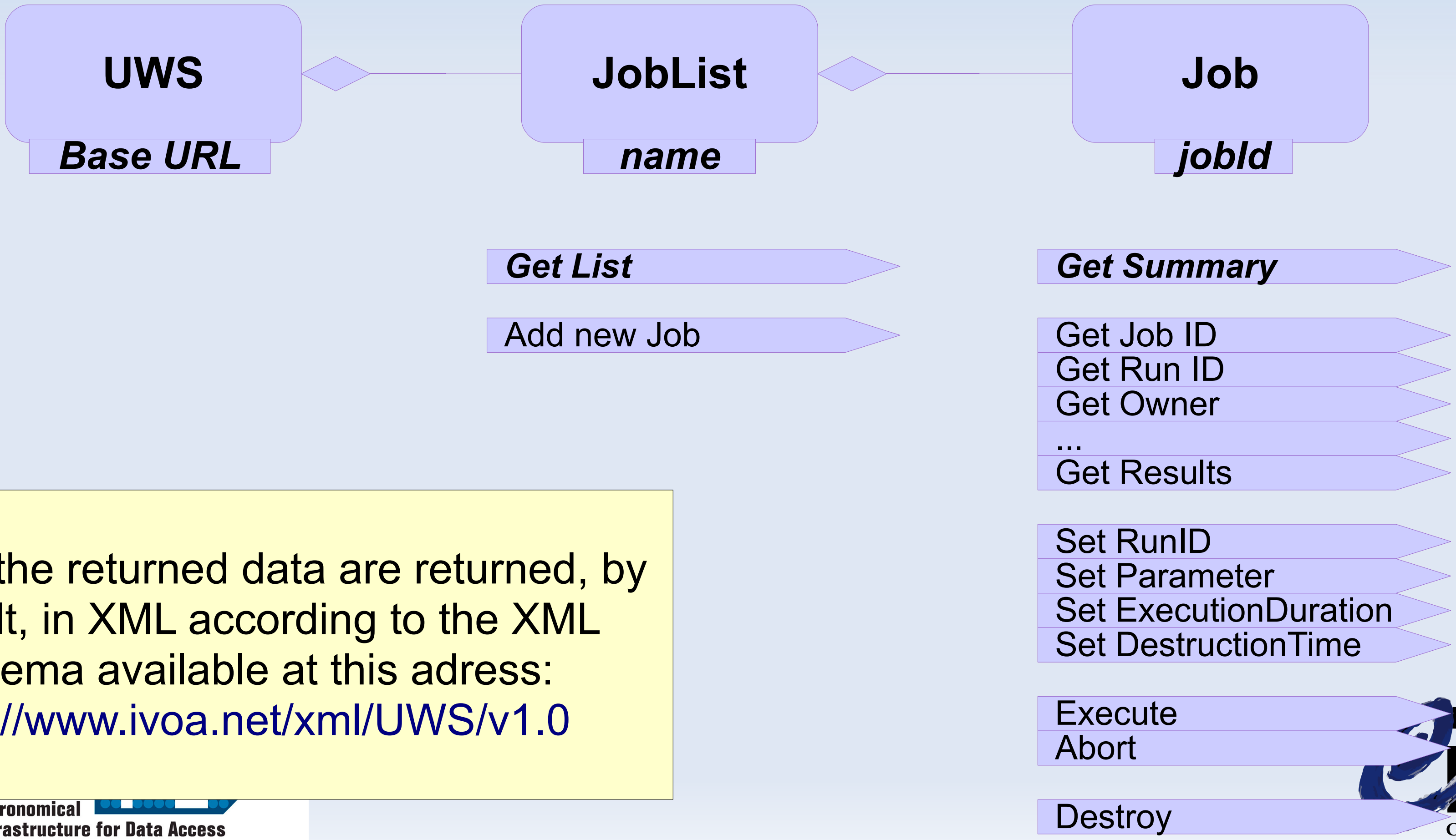


A job has information about its execution *(phase, start/end time, max. duration, results, error, ...)*

Quick description

UWS commands

- ...which can be manipulated thanks to some commands...



Most of the returned data are returned, by default, in XML according to the XML schema available at this adress: http://www.ivoa.net/xml/UWS/v1.0

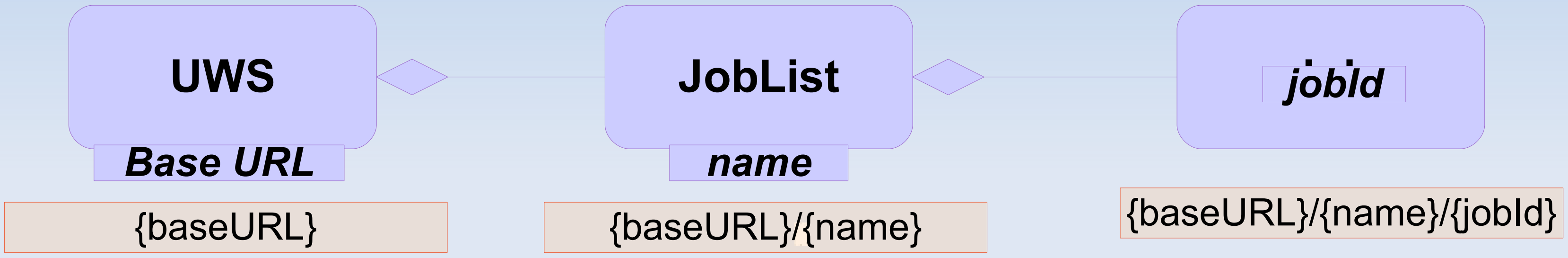
AIDA Astronomical Infrastructure for Data Access

CENTRE DE DONNÉES ASTRONOMIQUES DE STRASBOURG

Quick description

UWS URLs

- ...which correspond to REST based URLs.



For instance: (*if baseURL = http://foo.org, name = myJobList, jobId = 123Job*)

- **Get Job List** http://foo.org/myJobList in HTTP-GET
- **Add Job** http://foo.org/myJobList in HTTP-POST with/without some job parameters
- **Get Job Summary** http://foo.org/myJobList/123Job in HTTP-GET
- **Execute the job** http://foo.org/myJobList/123Job/phase in HTTP-POST with PHASE=RUN
- **Get Job Results** http://foo.org/myJobList/123Job/results in HTTP-GET
- ....

The UWS Library

Goal

- **Main Goal:**

  Providing a generic way to implement as quickly and as simply as possible the default behaviours of UWS.

- ***Notes:***
  - *Developed in Java*
  - *Designed to be used in Servlets*

Astronomical Infrastructure for Data Access

Centre de Données astronomiques de Strasbourg

The UWS Library

Functionalities

- **Implemented UWS functionalities described in standard:**
  - Interpreting each HTTP requests sent as UWS commands *(managed HTTP methods: GET, POST, PUT and DELETE)*
  - Stopping the job when its execution is longer than its imposed duration
  - Destroying the job at its imposed destruction time
  - Returning a UWS content in other formats than XML *(in version 3)*
  - Managing a job execution queue *(in version 3)*
- **Additionnal functionalities are also available:**
  - Customizing the UWS home page *(accessible via {baseURL})*
  - Linking each returned XML with a XSLT style-sheet
  - Adding custom commands to a UWS *(in version 3)*

EURO VO AIDA Astronomical Infrastructure for Data Access

CENTRE DE DONNÉES ASTRONOMIQUES DE STRASBOURG

The UWS Library

Quick HOW TO 1/2

To create your own job, you must:

1. Extend AbstractJob
2. Implement:

➔ **JobWork():**	*what the job must do*
➔ IsQueuedRequired():	*whether this job can be managed in a queue*

<u>Example:</u> JobChrono (a job which stops after a given number of seconds):

http://saada.u-strasbg.fr/uwstuto/gettingStarted.html#jobChrono

EURO VO AIDA Astronomical Infrastructure for Data Access

CENTRE DE DONNÉES ASTRONOMIQUES DE STRASBOURG

The UWS Library

Quick HOW TO 2/2

To create a UWS, you must:

1. Create a HttpServlet
2. Override the *doGet* and *doPost* functions
3. In *doGet*:
   i. Call the *doPost* function
4. In *doPost*:
   i. At the first call, initialize your UWS
   ii. Otherwise, call the function *executeRequest* of your UWS instance

<u>Example:</u>* *UWSTimers (a UWS which manages instances of JobChrono):

http://saada.u-strasbg.fr/uwstuto/gettingStarted.html#servletTimers

EURO VO AIDA Astronomical Infrastructure for Data Access

CDS Centre de Données astronomiques de Strasbourg

The UWS Library

Download/Tutorial

- Download:

  http://saada.u-strasbg.fr/saada/spip.php?article219
- Released under LGPL3 licence

- Documentation/Tutorial at:

  http://saada.u-strasbg.fr/uwstuto/

- Some answers or suggestions ?

  gregory.mantelet@astro.unistra.fr

EURO VO AIDA Astronomical Infrastructure for Data Access

CENTRE DE DONNÉES ASTRONOMIQUES DE STRASBOURG

Suggestions for the UWS pattern

- Add an attribute *progression* to a Job
  - Readable at any time
  - Writable by the job only during its execution
- Allows for multiple jobLists
- Add a new resource: *uws*
  - it has, optionnaly, a name and a description
  - it gives a list of job lists
- Propose a structure for a JSON format

EURO VO AIDA Astronomical Infrastructure for Data Access

CENTRE DE DONNÉES ASTRONOMIQUES DE STRASBOURG

Example of the XML content of *uws*

```
<uws name="uwsName">
   <description>...</description>
   <jobLists>
      <jobListRef name="jlName"
       href= ".../uwsName/jlName" />
      ...
   </jobLists>
</uws>
```

Services using the library

- Used in SAADA (ObsTAP implementation)
- Used in CDS cross-match service

Astronomical Infrastructure for Data Access

Centre de Données astronomiques de Strasbourg

Developer's feedback

Using the UWS library for the CDS X-match service

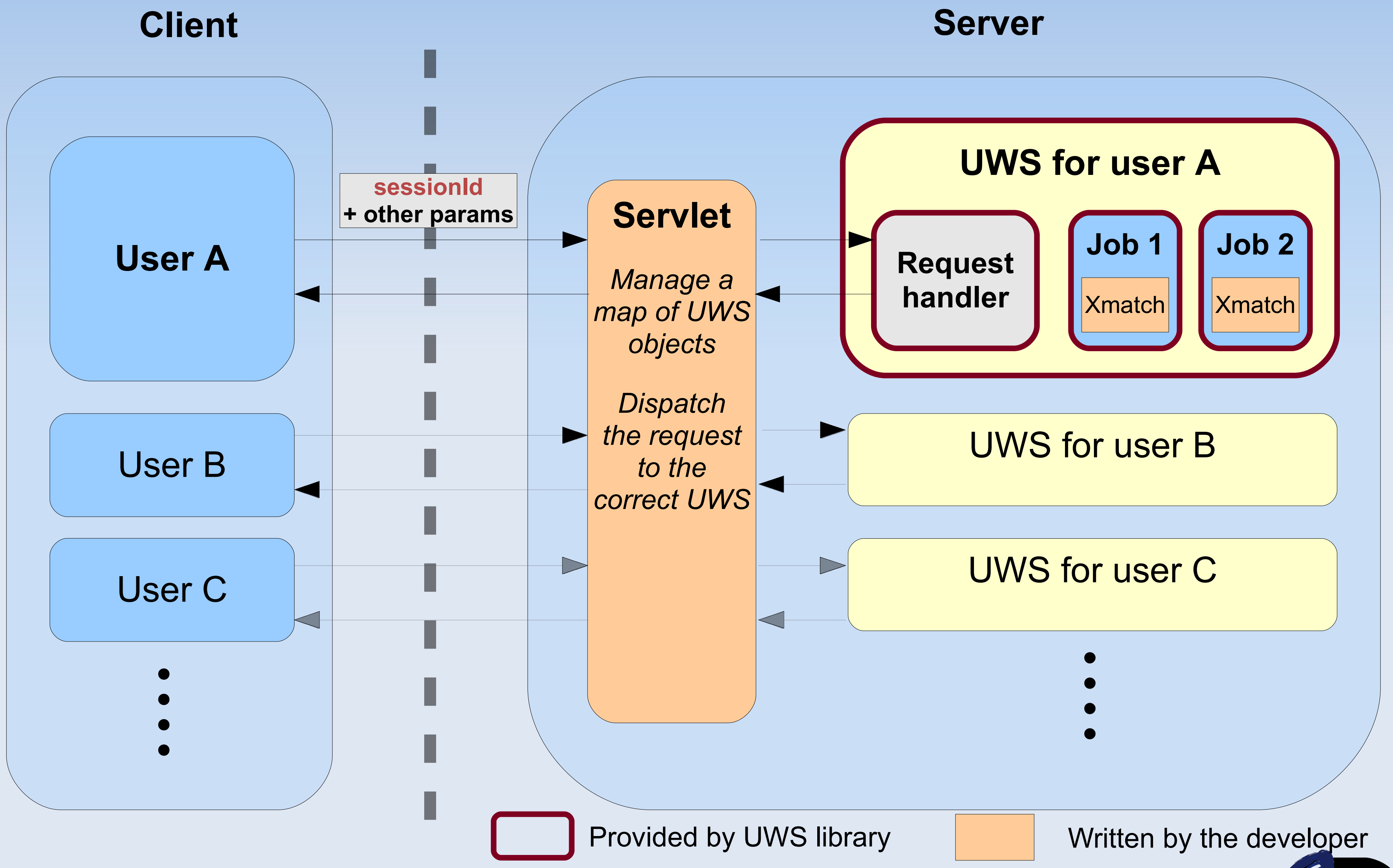
Thomas Boch

On behalf of: ***Brice Gassmann***

EURO VO AIDA Astronomical Infrastructure for Data Access

Centre de Données astronomiques de Strasbourg

UWS library for X-match service



Centre de Données astronomiques de Strasbourg

UWS library for X-match service

Thanks to the library, enabling UWS on the X-match service was seamless

Only 2 classes to extend

- *AbstractUWS* to associate a user id to a UWS
- *AbstractJob* to manage the additional parameters and perform the X-match itself

...and a servlet to write

- Manage a <userId, uws> map
- Dispatch the requests to the correct UWS objects (one UWS instance per user)

UWS library for X-match service

Easy interaction between user Web interface and the service

- UWS is REST based => perfect for AJAX requests
- But: JSON is not handled yet

Possibility to plug specific ”actions” to extend inner commands of UWS

- *deleteJobs* to delete a selection of jobs
- *getJobs* to get the list of jobs in JSON format
  (should we use HTTP Accept header to manage this ?)

EURO VO AIDA Astronomical Infrastructure for Data Access

Centre de Données astronomiques de Strasbourg

Dealing with multiple users

- Current implementation creates one UWS instance per user
  - Not optimal
  - Would be prettier if a *userId* or *sessionId* could be passed as a parameter of **/jobList**

Open questions

How do we isolate different users ?

- User A should not be able to see jobs from user B
- UWS document has a section about Security considerations (authentication/authorization)
  - Does it also apply to privacy ?
- Our implementation use session IDs to isolate users' jobs
  - Good pratice for privacy ?
  - Is there a proper (standardized) way to do that ?

JSON representation of objects

Suggested JSON format

```
*** UWS ***
{
	"name": "uwsName",
	"description": "uwsDescription",
	"jobLists": [
		{ "name": "jlName", "href": "jlUrl" },
		...
	]
}

*** JobList ***
{
	"name": "jlName",
	"jobs": [
		{ "id": "jobId", "href": "jobUrl", "phase": "jobPhase" },
		...
	]
}
```

EURO VO AIDA Astronomical Infrastructure for Data Access

CENTRE DE DONNÉES ASTRONOMIQUES DE STRASBOURG

```
*** Job ***
{
"jobId": "",
"runId": "",
"owner": "",
"phase": "",
"quote": "",
"executionDuration": "",
"destruction": "",
"startTime": "",
"endTime": "",
"error": { "type": "", "hasDetail": "", "message": ""},
"parameters": [
{ "paramName": "", "paramValue": "" },
...
],
"results": [
{ "id": "", "type": "", "href": "" }
...
]
}
```