



*International*

*Virtual*

*Observatory*

*Alliance*

## Introduction to CEA and UWS

### Version 1.00

*IVOA Note 2007 October 05*

**This version:**

CEAIntroduction-2007-10-05

**Previous version(s):**

None.

**Author(s):**

Guy Rixon

---

### Abstract

Discussion of the Common Execution Architecture and the Universal Worker Service involves a large number of technical terms for similar and overlapping concepts. This Note defines those terms and attempts to show the similarities and differences between concepts. There is a historical summary of the ideas; a glossary of terms; introductions to two key concepts; and a summary of the relevant standards in IVOA.

### Status of This Document

This is a Note.

*This is an IVOA Note expressing suggestions from and opinions of the authors. It is intended to share best practices, possible approaches, or other perspectives on interoperability with the Virtual Observatory. It should not be referenced or otherwise interpreted as a standard specification.*

A list of [current IVOA Recommendations and other technical documents](http://www.ivoa.net/Documents/) can be found at <http://www.ivoa.net/Documents/>.

## Contents

<a href="#">1 Origins.....</a>	<a href="#">2</a>
<a href="#">2 Glossary.....</a>	<a href="#">3</a>
<a href="#">3 Asynchronous control.....</a>	<a href="#">4</a>
<a href="#">3.1 Synchronous, stateless services.....</a>	<a href="#">4</a>
<a href="#">3.2 Some IVO activities that outgrow synchronous, stateless services.....</a>	<a href="#">5</a>
<a href="#">3.3 Asynchronous and stateful services.....</a>	<a href="#">6</a>
<a href="#">4 Wrapped and registered applications.....</a>	<a href="#">6</a>
<a href="#">5 UWS vs. UWS-PA vs. CEA.....</a>	<a href="#">7</a>
<a href="#">6 CEA v1 and v2 compared.....</a>	<a href="#">7</a>
<a href="#">7 Improvements in UWS-PA w.r.t CEC.....</a>	<a href="#">7</a>
<a href="#">8 Standards and other documents.....</a>	<a href="#">8</a>
<a href="#">9 References.....</a>	<a href="#">8</a>

## 1 Origins

In 2002, AstroGrid was looking for a way to run data-processing applications as services. Rather than defining and implementing a service contract for each application, of which there were many, AstroGrid wanted to write code that would be common to all applications.

At the time there was little prior art. Grid toolkits of the day were tried and found wanting. IVOA had standards for data access but none for data processing. Therefore, AstroGrid developed a new framework that used the IVOA registry standards. Over the next two years this framework evolved into v1 of the Common Execution Architecture (q.v.).

CEA solves a general problem: it is a service standard for “everything else” after IVOA has standardized particular applications such as image access and database queries. In the process, CEA solves the meta-problem of asynchronous control of jobs on services (q.v.).

CEA v1 was later adopted as a standard of EuroVO. It was proposed to IVOA [1] as a standard but was not immediately adopted. Conversion of the proposal to a full, IVOA standard was expected but not actively pursued.

Meanwhile, the Grid movement produced first the Open Grid Service Infrastructure [2] and then the Web-Service Resource Framework standards [3] which solve independently the asynchronous-control problem. Neither OGSi nor WS-RF are a complete replacement for CEA v1, but both have features that would improve CEA. Therefore, the Grid and Web Service working group of IVOA undertook to make a new version of CEA incorporating these new ideas. The key standards are: UWS (q.v.), the pattern for asynchronous control; UWS-PA, the worker service for CEA v2; Application resources for the IVOA registry. UWS has applications outside CEA.

In March 2007, EuroVO (VOTech) agreed to adopt CEA v2 and to support the development of its IVOA standards.

## 2 Glossary

CEA: the Common Execution Architecture. It consists in a contract for a web-service; XML schemata describing jobs (q.v.) passed to this kind of service; further XML schemata describing registrations of the web services and, separately, the wrapped applications (q.v.).

CEA: used loosely inside AstroGrid to indicate an application-server (q.v.) web-application: “we deployed a CEA at Portsmouth”. This usage is deprecated.

Application server: in the context of CEA, a web application that allows users to run “wrapped” (q.v.) applications. Application servers offer the CEC interface (q.v.) or the UWS-PA interface (q.v.) or both.

UWS (IWOA standard): Universal Worker Service pattern. This defines asynchronous control (q.v.) of long-running jobs (q.v.) on web services. UWS is a pattern rather than a complete contract for a service: it needs a specific application to define the details of inputs and output. UWS-PA (q.v.) is one such application.

UWS-PA: Universal Worker Service for Parameterized Applications. UWS-PA is an application of the UWS pattern (q.v.). It defines a service that can run wrapped applications (q.v.) which are individually registered in the IVO registry.

Wrapped application: an algorithm or code arranged to run inside an application server (q.v.). The application is typically legacy code. In the ideal case it is a well-known, piece of astronomical software with supporting journal-papers. The wrapping process should make the least possible change to the thing wrapped; it should certainly not alter the algorithms or the scientific details.

CEC: Common Execution Connector, a contract for the SOAP interface of an CEA application-server (q.v.). CEC is the kind of web service used in CEA v1 (q.v.).

Asynchronous control of jobs: a pattern for client-server interactions where the client submits a job to the service and then breaks communication while the job is executed; the client later polls the service to discover the progress of the job. This approach has also been called “non-blocking calls to the service”. C.f. synchronous control of jobs (q.v.).

Long-running job: in the context of CEA and UWS, any job that potentially takes longer than a standard HTTP timeout (typically 200 seconds). Long-running jobs need asynchronous control (q.v.).

OGSI: Open Grid Services Infrastructure, a standard from the Global Grid Forum. OGSI informed the development of UWS (q.v.).

WS-RF: Web-service Resource Framework (ref), a standard of OASIS (ref). WS-RF replaced OGSI in the Grid movement. It helped to define the lifetime-management in UWS.

CEA v1: first version of the Common Execution Architecture as used by AstroGrid and EuroVO from 2002 to 2007. CEA v1 involves the CEC (q.v.) interface.

CEA v2: revised version of the Common Execution Architecture developed in 2006-2007 by EuroVO and IVOA. CEA v2 uses the UWS (q.v.) and UWS-PA (q.v.) standards.

## 3 Asynchronous control

### 3.1 Synchronous, stateless services

Simple web services are *synchronous* and *stateless*. Synchronous means that the client waits for each request to be fulfilled; if the client disconnects from the service then the activity is abandoned. *Stateless* means that the service does not remember results of a previous activity (or, at least, the client cannot ask the service about them).

Synchronous, stateless services work well when two criteria apply.

1. The length of each activity is less than the “attention span” of the connection.
2. The results of each activity are compact enough to be easily passed back to the client via the connection on which the request was made (and possibly pushed back to the service as parameters of the next activity).

There are various limits to the attention span.

1. HTTP assumes that the start of a reply quickly follows its request, even if the body of the reply takes a long time to stream. If the service takes too long to compute the results and to start the reply, then HTTP times out at the request is lost.
2. A client runs computer which will not stay on-line indefinitely.
3. A network with finite reliability will eventually break communications during an activity.
4. A service is sometimes shut down for maintenance.

Synchronous, stateless services, in short, do not scale well.<sup>1</sup>

### 3.2 Some IVO activities that outgrow synchronous, stateless services

These cases are examples. They are not a complete list!

---

<sup>1</sup> Matthew Graham points out that it is possible to build a scalable system entirely from stateless, synchronous services by careful choice of architecture. However, I maintain that such architectures are not simple (no simpler than UWS), and a service protocol that looks attractive for light-weight use becomes onerous once when warped into a scalable architecture. Further, scalable, stateless architectures generally rely on continuous coordination by a client application; they do not adapt well to batch processing.

1. An ADQL service gives access to a large object-catalogue. Most queries run in less than a minute, but some legitimate queries involve a full-table traverse and take hours or days. The service needs to run these special cases in a low-priority queue.
2. An object-finding service runs the SExtractor application on a list of images. Normally, the list is short and the request is quickly satisfied. Occasionally, a list of 10,000 images is sent in the expectation that the work will be finished over the weekend.
3. A cone-search request on a rich catalogue raises 10,000,000 rows of results, but the client is connected via a slow link and cannot read all the results in a reasonable time. The client needs the service to send the results into storage over a faster link. This could mean sending them to VOSpace, or simply holding them temporarily until the user can retrieve them on a fast link.
4. An ADQL service allows users to save query results into new tables such that they can be the target of later queries. However, space is limited and the results tables can only be kept for a short time. The client and service negotiate the lifetime of the results tables.
5. A service performs image stacking on a list of fields. Each field can be processed by a synchronous service but the list is long and the user wants to retrieve the results of the early fields before the last fields are processed.

### 3.3 *Asynchronous and stateful services*

Services can be made to scale better by making them *asynchronous* and *stateful*. Asynchronous means that a client makes two or more separate requests to the service in the course of one activity, and that the client and service may be disconnected, possibly for days or more, in between those requests. Stateful means that the service stores state information about the activity and the client addresses requests to this state.

Web services that are asynchronous are almost always stateful. Most of special extra arrangements for asynchronous activities are actually managing the state of the activity.

There is an important class of stateful services where the state is peculiar to one job or session and the job is “owned” by one user. These, for the purpose of this document, are called *job-oriented* services. There are stateful services that are not job-oriented (e.g. a service managing a shared, client-writable DB table), but UWS does not apply to these.

For the purpose of this discussion, let the term *job* refer to the work specified by the JDL instructions and the term *resource* refer to the state of the job as recorded by the service. Both have a finite duration. The *lifetime* of the resource – i.e. the time from inception until the service forgets the state – is generally finite and must be at least as long the duration of the job.

## 4 Wrapped and registered applications

In CEA terminology, “wrapping” an application means making it capable of execution by an application server. This application server is a web service, implementing CEA protocols, that runs applications as jobs.

In AstroGrid's implementation of CEA v1, wrapping means defining the application's interface in a configuration document. The XML vocabulary of this document need only be understood by a particular service implementation; the document is not directly exposed to CEA clients.

For some legacy applications the wrapping also requires a shell script. The script is called by the application server and in turn calls the application. Such a script may handle parameters in a way not covered by the application server's configuration language. The script may also apply post-processing such as converting an ASCII-table output into VOTable.

Registering a CEA application means describing it in an IVO resource document and entering that document into an IVO registry. Registration allows a CEA client application to discover a application and immediately to generate a UI for invoking that application; i.e. a CEA client never needs a new software release to allow access to a new application.

IVOA's XML schema for registering CEA applications [4] is currently under development and will be based on VOResource 1.0. For CEA v1, AstroGrid currently uses an older schema [5] based on VOResource 0.10.

## 5 UWS vs. UWS-PA vs. CEA

UWS [6] is a pattern for asynchronous control of a job. It can be embedded in a number of service protocols the implementations of which can then share some code.

To apply the UWS pattern to a particular protocol, one has to specify the details of the JDL (i.e. how requests are posted to the service) and the details of the results (i.e. the set of resources that may be read after the job completes).

The proposed service-protocol for submitting and managing jobs in CEA v2 is “Universal Worker Service for Parameterized Applications” (UWS-PA). It replaces the Common Execution Connection (CEC) protocol of CEA v1.

There are applications of UWS outside CEA. For example, the second-generation DAL services are specified to use UWS for controlling long-running jobs.

## 6 CEA v1 and v2 compared

CEA v1	CEA v2
Service interface: CEC (q.v.).	Service interface: UWS-PA (q.v.).
Registrations based on VOResource v0.10.	Registrations based on VOResource v1.0.
Astrogrid/EuroVO standards.	IVOA standards.
Used within AstroGrid and EuroVO; little use elsewhere.	Likely to be ubiquitous in EuroVO; possibly used throughout the IVO.

## 7 Improvements in UWS-PA w.r.t CEC

UWS-PA (q.v.) is a replacement for CEC (q.v.). It adds a number of features.

- Control of job lifetime (q.v.).
- Services “quoting” (q.v.) for the time to complete a job.
- Access to a job's “sandbox” (q.v.) in order to inspect intermediate files.
- Access to VOspace, replacing AstroGrid MySpace.

CEC defines a SOAP service; UWS-PA defines a REST service. It is expected that UWS-PA will be much easier to implement and to call from clients.

## 8 Standards and other documents

CEA v1 (q.v.) is an internal standard of AstroGrid that has been adopted by EuroVO. The standard is described by an IVOA note [1] and the AstroGrid implementation is described in AstroGrid's architecture-document [7]. The details of the standard consist in a set of XML schemata published by AstroGrid.

CEA v2 (q.v.) is an emerging standard of EuroVO. Its component parts are, or will become, IVOA standards. UWS (q.v.) has an IVOA draft-standard [6]; UWS-PA (q.v.) will soon have one. The schema by which a wrapped application (q.v.) can be registered is a draft standard of IVOA [5].

## 9 References

[1] P. Harrison, *A Proposal for a Common Execution Architecture*, IVOA note May 2005, <http://www.ivoa.net/Documents/latest/CEA.html>

[2] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling, P. Vanderbilt, *Open Grid Services*

- Infrastructure (OGSI)*, pub Global Grid Forum 2003,  
[http://www.globus.org/toolkit/draft-ggf-ogsi-gridservice-33\\_2003-06-27.pdf](http://www.globus.org/toolkit/draft-ggf-ogsi-gridservice-33_2003-06-27.pdf)
- [3] OASIS web service resource framework technical committee, WSRF public documents, [http://www.oasis-open.org/committees/documents.php?wg\\_abbrev=wsrf](http://www.oasis-open.org/committees/documents.php?wg_abbrev=wsrf)
- [4] AstroGrid, Application schema based on VOResource 0.10,  
<http://software.astrogrid.org/schema/vo-resource-types/CEAService/v0.2/CEAService.xsd>
- [5] P.Harrison, Application schema based on VOResource 1.0,  
<http://software.astrogrid.org/schema/vo-resource-types/CEAService/v1.0rc1/CEAService.xsd>
- [6] G. Rixon, *Universal Worker Service v0.3*, IVOA unpublished draft (for information only), <http://www.ivoa.net/internal/IVOA/IvoaGridAndWebServices/UWS-0.3.pdf>
- [7] G. Rixon, *Software Architecture of AstroGrid v1*,  
<http://software.astrogrid.org/developdocs/astrogrid-v1.1-architecture.htm>