# Tuple Space

## Purpose

To provide, through a standard web-service interface, a simplified tuple-space-like capability which can be used by other services and by applications running in the ELF container.

We have chosen to implement this service instead of using JavaSpaces, the tuple-space which has long been part of the Jini Technology package (see http://java.sun.com/developer/products/jini and also http://javaspaces.org), for the following reasons:

1. There are serious problems with using multicast technologies on NCSA's production resource subnets.
2. We do not require the full peer-to-peer capabilities offered by Jini.
3. We do not require the full generality of JavaSpaces with respect to serializable classes; our service accepts an open-ended (i.e., basically untyped) wrapper object containing as its payload element a serialized XML blob (which usually, but not necessarily, will implement `ncsa.tools.common.UserFacing`).

## API

### Service WSDL

### ITupleSpace

```
public interface ITupleSpace
{
        Integer write( String entryXML ) throws Throwable;
        Integer[] writeB( String[] entryXML ) throws Throwable;
        String[] read( String matchXML, int max ) throws Throwable;
        String[] take( String matchXML, int max, long timeout ) throws Throwable;
        String[] takeNB( String matchXML, int max ) throws Throwable;
}
```

#### NOTES

- `write` takes a serialized string representing the `ncsa.service.vizier.tspace.TspaceEntry` object (see below); returns the entry id.
- `writeB` is a batch method which does the same thing as `write` on multiple entries; returns array of ids corresponding to each entry.
- `read` and `take` are batch methods, both requiring a serialized string representing the `ncsa.service.vizier.tspace.TspaceMatchRequest` object (see below), along with an integer indicating the maximum number of entries to return (default is 10; it is advisable not to set this number too high, as Axis will have trouble managing extremely large SOAP packets); the array returned contains serialized `ncsa.service.vizier.tspace.TspaceEntry` objects.
- `take` also requires a timeout (if set to t <= 0, the non-blocking method is called), whereas `takeNB` will return immediately subsequent to issuing the request against the data store.
    - `takeNB` can return `null` if no entries satisfy the request constraints.
    - `take` will not return `null` unless timeout <= 0.
    - `take` may return an empty array (length 0) if its timeout has been exceeded.

### `ncsa.service.vizier.tspace.TspaceEntry`

The object stored in the service's data store and which can be retrieved through the `read` or `take` ports. A fuller description of the object is found on the ncsa.tools.ogrscript.tspace page.

#### NOTE

When passing this object to the `write` port, it is important that the `id` attribute/field be handled correctly.

- If `id` is `null`, the object will be added as a new entry.
- If `id` is not `null`, an attempt will be made to update an existing object with that id. The fields subject to update are:
    - *timeTaken*
    - *timeCompleted*
    - *retried*
    - *ranOn*

If the `id` is set, it is not necessary to provide a complete description of the entry; only the XML elements or attributes corresponding to the 4 updateable fields above will be processed. See further the ncsa.tools.ogrscript.tspace page.

### `ncsa.service.vizier.tspace.TspaceMatchRequest`

The object used to express the match query parameter required by the `read` or `take` ports. A fuller description of the object is found on the ncsa.tools.ogrscript.tspace page.