Suggested format for IVOA validation response

V0.0 May 11, 2016

1. Typical Use Cases

- IVOID identifier validation
     Checks whether an input string is syntactically valid and present in registry.

- Single cone search service
    Does a sequence of checks of both correct and invalid inputs on a single base URL.

- TAPlint on a single service
    Runs a series of distinct sets of tests on a single TAP service.

- Validation of a set of services
    Runs validation on a set of distinct services (possibly implementing distinct protocols)

2. Conceptual framework for output.

Represent the output as a hierarchical structure of arbitrary depth allowing for grouping of tests appropriate to a given use case while minimizing duplication by allowing test characteristics to be defined at a higher group level.  This can be implemented using either a JSON or XML format but we should probably pick.  The basic Test structure has the following elements:

**name**: A string identifying the test.  Required.
**protocol**: The protocol being tested. This should normally be available but may be inherited from a higher level.  We should create a controlled vocabulary for this.

**validator**: An identifier for the validator being used. Normally present only at the top level of the output, but in principle results from multiple validators could be combined. Optional.

**protocolSection**:  A pointer to a specific section of the protocol.  Optional.

**input**: The inputs to the test.  Input is a structure with the following fields:

**baseURL**: A base URL to the service being tested
**value**:  A string value to be tested.

One of baseURL or value should be specified but may be inherited from a higher level.
**parameters**: one or more keyword/value pairs specifying specific parameters to the test. These would normally be appended to the base URL.

An input structure should normally be given, but may be inherited by lower levels in the test.
An input structure in a lower level may only specify parameters and still inherits the baseURL.

**status**: A result of the test in some TBD controlled vocabulary.  This is normally expected for all tests which do not have subtests but it optional at higher levels.
          Suggested initial vocabulary:
                *Pass, Fail*      For specific binary tests
                *Correct*  -- no subtests fail
                *Working* -- failures in subtests are deemed non-critical

 *Skipped* – The test was skipped for some reason which should be given in resultText
                 *Failing* -- failures in subtests are deemed critical
                 *Broken* -- problems found in connection with service or fundamental problems in
                                  returned format.

**resultText**: Text describing the result.    Recommended especially for failing tests but optional.

**subTests**:  An array of 0 or more subtests associated with the test.  This is optional.

# Examples

Note that the examples below are designed to illustrate the format and currently only have a vague relationship with the actual validation services and protocols mentioned (e.g., the section numbers are made up). Examples are given in JSON and XML to illustrate what these might look like, but a final proposal should probably pick one.

## Validating an IVOID

{

 “validator” : ”DaCCHS IVOID Validator v1.2 rev127”,

 “protocol” : “IVOID 1.3”

 “name” : “Overall result”,

 “input” : {

 “value”: “ivo://nasa.heasarc/rosmaster”

 },

 “status” : “correct”,

 “subtests” : [{

 “name”: “syntaxvalid”,

 “status”: “pass”,

 “protocolSection”: “2.1,2.2,2.3”

 }, {

 “name”: “foundinregistry”,

 “status”: “pass”,

 “protocolSection”: “2.4”

 }]

}

## TAPLint validation (fragments)

<Test>

 <Name> TAPLint </Name>

 <Validator> TAPLint v1.4 </Validator>

 <Protocol> TAP v1.0 <Protocol>

 <Input>

 <BaseURL> <http://somehost.host/tapbaseURL> </BaseURL>

 <Parameters>

 <Tests> DAL,UWS </Tests>

 </Parameters>

 <Input>

 <Subtests>

 <Test>

 <Protocol> DALI 1.1 </Protocol>

 <Name> DAL </Name>

 <Subtests>

 <Test>

 <Name> DAL-AVAIL </Name>

 <Status> Fail </Status>

 <ResultText> Availability does not match controlled vocabulary </ResultText>

 <ProtocolSection> DALI 4.2 </ProtocolSection>

 </Test>

 <Test>

 <Name> DAL-CAPAB </Name>

 <Status> Pass </Status>

 </Test>

 …

 </Subtests>

 </Test>

 <Test>

 <Name> UWS </Name>

 <Protocol> UWS 1.2 </Protocol>

 <Subtests>

 <Test>

 <Name> UWS-CREATEJOB </Name>

 <Input>

 <Parameters>

 <Request>select \* from mytable</Request>

 </Parameters>

 <Status> Fail </Status>

 <ResultText> Unable to create query job for table </ResultText>

 </Test>

 ….

 </Test>

</Test>