Plan Management API

Authors

Person	Role	Partner	Contribution
Frank Asseg	DEV	FIZ	
Matthias Hahn	DEV	FIZ	

Distribution

Person	Role	Partner
PT Group		

Revision History

Version	Status	Author	Date	Changes
0.1		Frank Asseg	2012-05-08	First draft
0.2		Frank Asseg	2012-08-11	Update
0.3		Frank Asseg	2012-08-15	Update
0.4	Final	Frank Asseg	2012-09-21	3.7, 3.8, 3.9

Table of Contents

1. Introduction

The Plan Management API is a set of HTTP endpoints for serving content in a SCAPE environment. It's purpose is to integrate the various components in the SCAPE platform that handle preservation plans. First of all plans in use by the SCAPE platform are part of a digital objects provenance and therefore the need to persist these arises, so the repository acts as a storage system for preservation plans. Secondly the Plan Management API acts as a Bridge

in between the Workflow Execution Environment and a planning agent, relaying execution of plans as requested by the agent and supplying information about preservation plan state to the agent.

2. Use Cases of the Plan Management API

2.1. Deploying new plans in the Repository

An agent that created a new preservation plan using the Planning and Watch component needs to be able to put the new preservation plan into the repository, in order to have them execute on the workflow execution environment.

2.2. Get state information about plans

A Planning agent needs to be able to get state information about a preservation plan from the repository to monitor the result of preservation plan executions, and for deciding on eligibility of preservation plans for execution.

2.3. Change existing plans

If a plan is erroneous or the set of parameters for a given preservation plan has to be changed, it has to be updated in the repository. But since the provenance of a digital object may very well be referencing a preservation plan before it changed, a versioning system for preservation plans is required.

2.4. Get a specific plan based on some criteria

In the simplest case the agent has a preservation plan identifier and wants to fetch this plan for e.g. manipulation using Plato[1]. But also the agent might search for plans based on some plan properties like a description.

2.5. Reservation of Identifiers

In order to create new Plans the agent has to be aware of the identifier to use for the plan. Therefore a facility to request a reserved identifier is required, which is used when the preservation plan gest deployed on the repository.

2.6. Execute a preservation plan on the Workflow Execution Environment

An agent requires the ability to trigger plan execution on the Workflow Execution Environment.

3. Specification

3.1. XML Schemas

The PLATO XML schema describing the structure of a preservation plan is used for representing preservation plans in XML and can be found at:

https://github.com/openplanets/plato/blob/master/planning-core/src/main/resources/data/schemas/plato.xsd

The XML Schema for execution states will be made available as part of the scape-dto project which is currently under development on github at:

https://github.com/fasseg/scape-dto

[1] http://www.ifs.tuwien.ac.at/dp/plato/intro.html

3.2. Life cycle state of preservation plans

A preservation plan is associated with a life cycle state. This enables to differentiate in between e.g. active and inactive plans. The life cycle states used are as follows:

- ENABLED
  The preservation plan is ready for execution
- DISABLED
  The preservation plan is disabled and can not be executed

3.3. Execution status of preservation plans

Preservation plans can be associated with a state for every time the plan has been executed on the Workflow Execution Environment. Execution state consist of a timestamp, a description and one of the two following values:

- EXECUTION_IN_PROGRESS
  The preservation plan is currently being executed on the Workflow Environment
- EXECUTION_SUCCESS
  The preservation plan is currently running on the Workflow Execution Environment
- EXECUTION_FAIL
  The execution of the preservation plan failed and the plan is not available for execution until an agent changes this state back to ENABLED

Following is an example for the XML representation of an execution state describes a failure due to e.g. missing objects:

```
<execution-state timestamp="2007-11-05T14:30.2332" state="EXECUTION_FAIL">
    The following objects have not been updated: […]
</execution-state>
```

3.4. Authentication & Authorization

Following the REST[2] recommendations authentication is done by using Basic and Digest Access Authentication mechanism, also known as HTTP Basic Authentication, which is well established and supported throughout the otherwise heterogeneous IT landscape. This implies that every HTTP request has to have a BASE64 encoded user name and password string in the Authentication Header. Therefore encryption by using HTTP over SSL/TLS is strongly recommended.

Authorization is done by the repository depending on the current user and the individual Representations' associated rights and permission metadata.

3.5. HTTP Status codes

The existing HTTP status codes with their individual semantics are used in the context of the connector API.

[2] R.T. Fielding, 2000,"Architectural Styles and the Design of Network-based Software Architectures"

- **200 OK** This indicates success.
- **201 Created** means that an object has been created in the repository.
- **401 Unauthorized** The request requires authentication. The user should authenticate properly against the repository and repeat the request.
- **403 Forbidden** The server refuses to fulfill the request. Authorization will not help and the request should not be repeated.
- **404 Not Found** The requested resource cannot be found.
- **415 Unsupported Media Type** The media type sent with the requested was not valid**.**
- **500 Internal Server Error** In the case of runtime errors that might be happening while requesting a resource: e.g. Disk full.

3.6. HTTP endpoints

Following is a specification of the HTTP endpoints. The implementation of the endpoints has to be done by each repository in order to be deployed in a SCAPE Platform environment.

3.7. Retrieve a plan

An endpoint for plan retrieval is exposed by the repository. Plans are returned according to the Plato Version 4 XML schema, that is currently in development.

Path

/plan/<id>

Method

HTTP/1.1 GET

Parameter

id *the id of the plan to be fetched from the repository*

Produces

A XML representation of the plan

3.8. Deploy a new plan

An endpoint for deployment of new plans in the repository is exposed, via HTTP PUT. An agent can upload new preservation plans for later use or reference by sending a request structured as follows. An identifier for the plan can be requested via the method described in 3.13 before deploying.

Path

/plan/<id>

Method

HTTP/1.1 PUT

Consumes

A XML representation of the plan

3.9. Search plans

In order to be able to search plans based on their significant properties an endpoint for SRU searching is exposed by the repository. . The endpoint implements the SRU specifications by the Library of Congress for Internet Search queries, utilizing CQL, a standard syntax for representing queries, and exposes this functionality via a HTTP GET endpoint. Pagination is done via the SRU parameters

startRecord and *maximumRecords* [3]. A level 1 implementation according to the SRU standard is implemented in order to accommodate various use cases. A list of significant plan properties will be made available.

Path

/sru/

Method

HTTP/1.1 GET

Parameter

see SRU specification[4]

Produces

A URI list referencing the plans found by the search request

3.10. Retrieve plan execution states

In order to supply an Agent with feedback about the execution of preservation plans, the repository exposes and Endpoint for retrieving lists of execution states.

Path

/plan-execution-state/<id>

Method

HTTP/1.1 GET

Parameter

Id *the id of the preservation plan*

Produces

A XML representation of all the execution states associated with a preservation plan.

3.11. Add a plan execution state

The workflow execution environment needs the possibility to add new execution states to preservation plans, in order for this states to be available to the Plan Management GUI user.

Path

/plan-execution-state/<id>

Method

HTTP/1.1 POST

Consumes

A XML representation of the execution state

Parameter

[3] http://www.loc.gov/standards/sru/specs/search-retrieve.html

[4] http://www.loc.gov/standards/sru/

id *the id of the plan for which the execution state should be updated*

3.12. Update plan lifecycle status

An agent requires the possibility to change the life cycle state of a preservation plan to enable and disable specific preservation plans. Only enabled plans can be executed on the Workflow Execution Environment.

Path

/plan-state/<id>/<state>

Method

HTTP/1.1 PUT

Parameter

id *the id of the plan which is to be updated*

state *the new state of the plan.One of { ENABLED,DISABLED}*

3.13. Retrieve a reserved plan identifier

An agent can request a preservation plan identifier, which gets reserved by the repository. The reserved identifier has a 24h validity, so that a preservation plan using the reserved identifier has to be deployed within 24 hours or the identifier will be invalidated and unusable.

Path

/plan-id/reserve

Method

HTTP/1.1 GET

Produces

A XML representation of the Identifier reserved for up to 24h.

4. Glossary

A **preservation plan** consists of properties describing the plan, its purpose and its target objects . A preservation plan can be executed on the Workflow Execution Environment and will fetch and process digital objects which get updated via the Connector API.

Search/Retrieve via URL(SRU) is a standard search protocol for Internet search queries, utilizing Contextual Query Language(CQL), a standard query syntax for representing queries.

Representational state transfer (REST) is a style of software architecture for distributed hypermedia systems such as theWorld Wide Web. The term representational state transfer was introduced and defined in 2000 by Roy Fielding in his doctoral dissertation. Fielding is one of the principal authors of the Hypertext Transfer Protocol (HTTP) specification versions 1.0 and 1.1.