ws-prov-rec

Authors:
Paul Groth, U. Southampton
Simon Miles, U. Southampton
Victor Tan, U. Southampton
John Ibbotson, IBM
Luc Moreau, U. Southampton

August 24, 2006

The P-assertion Recording Protocol

Status of this Memo

This document provides information to the community regarding the specification of a protocol for recording process documentation into a provenance store and has the status of a working draft. The document does not define any standards or technical recommendations. Distribution is unlimited.

Copyright Notice

Abstract

Related documents [MGJ+06, TMG+06] define schemas to be used for documentation about the execution of a process, *process documentation*, and introduce a *provenance store*, a type of Web Service with the capability for storing and giving access to process documentation. In particular, process documentation has a defined schema, the *p-structure*, which clients use when creating process documentation to be recorded. In this document, we specify an interface, the P-assertion Recording Protocol (PReP) [GLM04], by which a recording actor can communicate with a provenance store in order to record process documentation. This primarily takes the form of an abstract WSDL interface defining messages to be accepted and produced by a provenance store.

Contents

1 Introduction

Every provenance store supplies a Web Service interface for recording process documentation. It has a single operation, *record*, that takes Record document as input and returns an acknowledgement as result. This document defines the schema for the request and acknowledgement messages. The WSDL 1.1 description of the interface is given in Appendix B.

1.1 Goals and Requirements

The goal of this document is to define a Web Service interface, the P-assertion Recording Protocol (PReP), for recording process documentation into a provenance store.

1.1.1 Requirements

In meeting this goal, this document must address the following requirements:

- Define the schema of the record request message sent to the provenance store.
- Define the behaviour of a provenance store on receiving a record request.
- Define the schema of the acknowledgement message returned by the provenance store.

1.1.2 Non-Requirements

This document does not specify any properties or guarantees that the protocol may have. Such a specification is left to an abstract definition of the messages required for recording process documentation [GLM04]. This document also does not define the storage format, medium, or technology that a provenance store uses.

2 Terminology and Notation

All definitions for the concepts and structures found within this document can be found in [TGJ$^+$06].

2.1 XML Namespaces

The XML Namespace URI that MUST be used by implementations of this specification is: `http://www.pasoa.org/schemas/version023s1/record/PRecord.xsd`

Table 1 lists XML namespaces that are used in this specification. The choice of any namespace prefix is arbitrary and not semantically significant.

Prefix	XML Namespace	Specification(s)
pr	`http://www.pasoa.org/schemas/version023s1/record/PRecord.xsd`	[PRecord]
ps	`http://www.pasoa.org/schemas/version023s1/PStruct.xsd`	[P-Structure]
xs	`http://www.w3.org/2001/XMLSchema`	[XMLSchema]

Table 1: Prefixes and XML Namespaces used in this specification

2.2 Notational Conventions

The keywords "MUST ", "MUSTNOT ", "REQUIRED ", "SHALL ", "SHALLNOT ", "SHOULD ", "SHOULDNOT ", "RECOMMENDED ", "MAY ", and "OPTIONAL " in this document are to be interpreted as described in [Bra97].

2.3 XML Schema Diagrams

This document adopts a graphical notation to describe XML Schema. Figure 1 gives an example of a small XML Schema displayed as a diagram, which is now explained with reference to the figure.

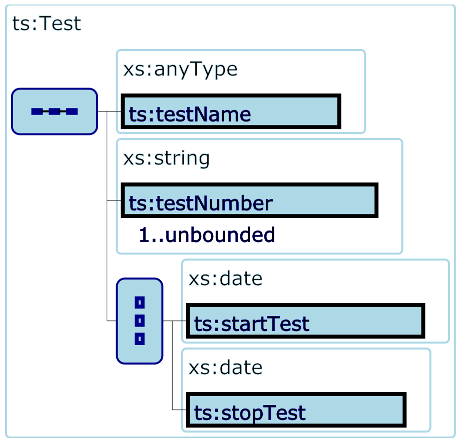
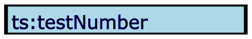


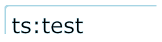
Figure 1: An example XML Schema diagram

Figure 1 defines the structure of type `ts:Test`. The type Test contains a sequence of elements, which we now detail. One element in the sequence is `ts:testName`, which can be any type and must occur once and only once in an instance of `ts:Test`. `ts:Name` is followed by element `ts:testNumber`, which must contain a string. The `ts:testNumber` element must occur at least once and can occur as many times as needed. This is denoted by the "1..unbounded" under the element. Finally, the sequence contains a choice between two elements, `ts:startTest` and `ts:stopTest`, either of which must contain a date.

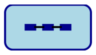
Below is a simple of description of each of the parts of the XML Schema diagram format.



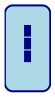
An element (instance) is represented by the qualified name of the element in the box. By default an element must occur once and only once. Where this restriction does not hold, the text “1..unbounded”, “0..unbounded”, “0..N”, “1..N” (where N is an integer) appears under the element box. The left hand number is the minimum occurrences of the element at the position in the XML document, the right hand number is the maximum (with “unbounded” for no maximum).



A complex type is denoted by a lightly marked box with the qualified name of the type at the top left. The structure of the type is given by the elements, types and control structures within the box.



A horizontal sequence of dots represents a sequence of elements or control structures, that must appear in an element conforming to the type in the surrounding type box.



A vertical sequence of dots represents a choice between elements or control structures, that must appear in an element conforming to the type in the surrounding type box.

2.4 XPath notation

In addition to the XML Schema diagrams, an XPath notation [W3C99] is used to identify each individual element in the specification along with its context, in order to describe precisely its meaning and use.

3 Recording Process Documentation

We specify below the request document schema, provenance store behaviour and acknowledgement document schema for recording process documentation. The full schema document, in which request and acknowledgement message structures are defined, is given in Appendix A. The WSDL 1.1 description of the interface taking and producing these messages is given in Appendix B. These schemas are based on those defined in the p-structure data model [MGJ+06], therefore, a familiarity with that document and the concepts it defines is assumed.

3.1 Request

A process documentation record request is a message sent by a *recording actor* to a *provenance store* that contains process documentation to be stored by the provenance store. It is instantiated as a document conforming to the schema depicted in Figure 2.

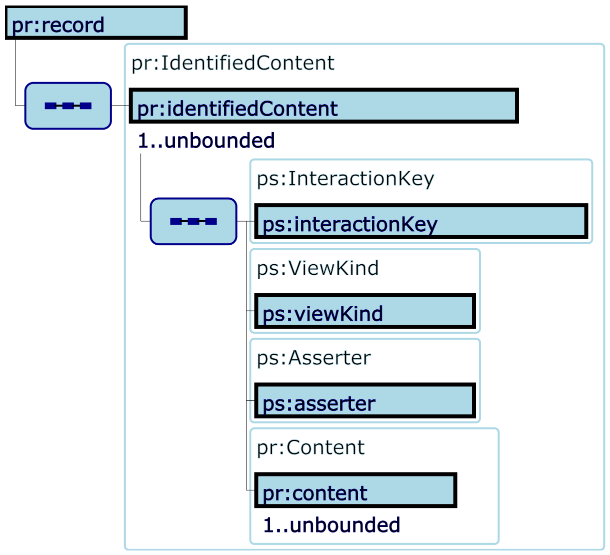


Figure 2: Process Documentation Record Request

One feature of the request message is to allow for bulk recording of multiple p-assertions and other associated information. For example, an actor may wish to record p-assertions at a time when network traffic is low or when the actor is not busy performing other tasks. Therefore, multiple pieces of information, which we call *content*, can be recorded about different interactions at the same time. Allowing content to be bundled together gives greater flexibility to the actor.

`/pr:record`

This element contains a record request.

`/pr:record/pr:identifiedContent`

This element is a container that associates a content element with the View that the content belongs to. There can be multiple `pr:identifiedContent` elements within a record request.

`/pr:record/pr:identifiedContent/ps:interactionKey`

This element identifies the key of the InteractionRecord that the content belongs to.

`/pr:record/pr:identifiedContent/ps:viewKind`

This element is the name of the View within the InteractionRecord that the content belongs to.

`/pr:record/pr:identifiedContent/ps:asserter`

This element is the identity of the asserter of the content.

`/pr:record/pr:identifiedContent/pr:content`

This element contains the information being recorded. This element is defined as a choice between the elements defined in Figure 3. This is to prevent any incorrectly typed information from being recorded. There can be multiple content elements within one `pr:identifiedContent` element.

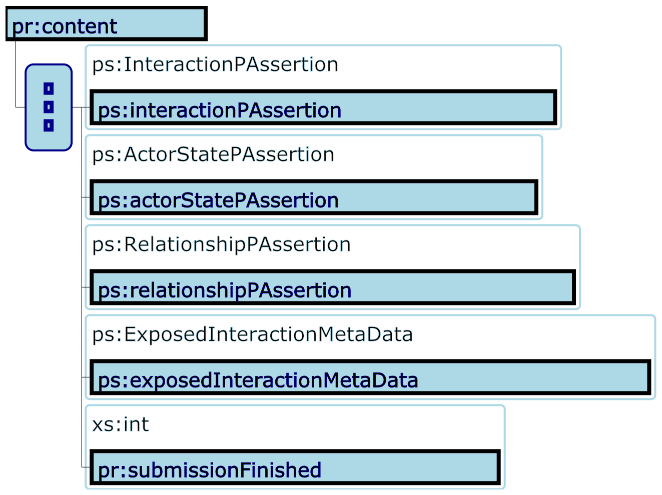


Figure 3: Content element definition

`/pr:record/pr:identifiedContent/pr:content/ps:interactionPAssertion`

An interaction p-assertion.

`/pr:record/pr:identifiedContent/pr:content/ps:actorStatePAssertion`

An actor state p-assertion.

`/pr:record/pr:identifiedContent/pr:content/ps:relationshipPAssertion`

A relationship p-assertion.

`/pr:record/pr:identifiedContent/pr:content/ps:exposedInteractionMetaData`

Some exposed interaction metadata.

`/pr:record/pr:identifiedContent/pr:content/pr:submissionFinished`

This element allows an actor to record how many p-assertions it expects to record in total. This allows querying actors to determine whether or not an actor is finished recording p-assertions.

3.2 Behaviour

On receiving a record message, a provenance store must store the contents of the message. The provenance store must return an acknowledgement, which may be sent synchronously or asynchronously depending on the underlying transport mechanism being used between the client and the provenance store. The provenance store must make the contents of the message available through the navigation of the p-structure via some query interface.

3.3 Acknowledgements

A record acknowledgement is sent by a provenance store to the actor that issued the corresponding record request. It is a document conforming to the schema depicted in Figure 4. Depending on the transport mechanism being used either a synchronous or asynchronous acknowledgement may be returned. The schema for the acknowledgement is defined here.

`/pr:recordAck`

This element contains a record acknowledgement. Each `pr:identifiedContent` element recorded has a corresponding acknowledgement element.

`/pr:recordAck/pr:synch_ack`

If the underlying transport is synchronous this element is used. The element has no internal data because the acknowledgement immediately follows the record request on the same communication channel, therefore, the sender already knows what record request `pr:recordAck` is acknowledging.

`/pr:recordAck/pr:ack`

This element is used when returning an acknowledgement asynchronously. In this case, clients may receive acknowledgements in any order, therefore, some means must be provided for clients to identify which record request each acknowledgement is associated with. Hence, several identifiers are provided that allow clients to make this association. These identifiers are defined below.

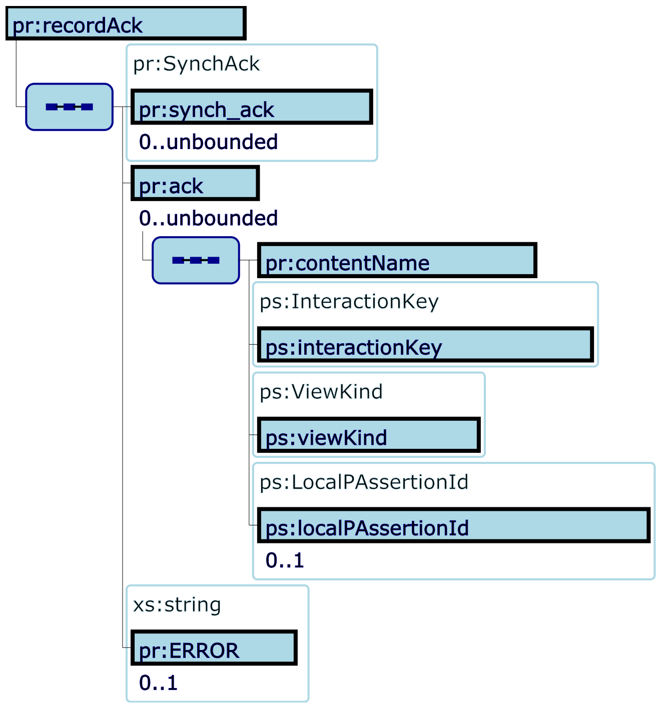


Figure 4: Process Documentation Record Acknowledgement

`/pr:recordAck/pr:ack/pr:contentName`

This element states the type of content that was recorded. The following standard strings are used, which map to element names defined in the `pr:content` element:

- interactionPAssertion
- actorStatePAssertion
- relationshipPAssertion
- exposedInteractionMetaData
- submissionFinished

Therefore, if a `pr:identifiedContent` element in the record request message contained an interaction p-assertion then the acknowledgement would contain a `pr:contentName` with the string interactionPAssertion as its contents.

`/pr:recordAck/pr:ack/ps:interactionKey`

The interaction key of the `pr:identifiedContent` element being acknowledged.

`/pr:recordAck/pr:ack/ps:viewKind`

The view kind of the `pr:identifiedContent` element being acknowledged.

`/pr:recordAck/pr:ack/ps:localPAssertionId`

If the recording of a p-assertion is being acknowledged, this is its local p-assertion id.

3.4 Faults

The acknowledgement message provides an element in which provenance stores can put implementation specific error messages.

`/pr:recordAck/pr:ERROR`

This element holds implementation specific error messages. This element may be present in both synchronous and asynchronous acknowledgement messages.

4 Security Considerations

This specification defines the process documentation record request and acknowledgement messages for any provenance store supporting the PReP protocol. In this context, there are two categories of security aspects that need to be considered: (a) securing the message exchanges and (b) securing the provenance store contents.

4.1 Securing Message Exchanges

When messages are exchanged between a recorder and provenance store when recording process documentation, it is recommended that the communication be secured using the mechanisms described in WS-Security [Var04]. In order to properly secure messages, the message body (record document or acknowledgement) and all relevant headers need to be included in the digital signature so as to prove the integrity of the message. In the event that a recorder frequently records process documentation, it is recommended that a security context be established using the mechanisms described in WS-Trust [Var05b] and WS-SecureConversation [Var05a], allowing for potentially more efficient means of authentication.

4.2 Securing Provenance Store Contents

Since this specification defines a mechanism to record process documentation in provenance stores, security policies should be established that ensure that only authorized actors can record p-assertions. A detailed architecture for securing provenance stores and their contents can be found in [GJ$^+$06].

5 Conclusion

In this document, a protocol for recording process documentation into a provenance store is defined. The definition includes the format of request and acknowledgement messages and the expected behaviour of provenance stores with respect to those messages. Using the protocol an actor can record process documentation compatible with the p-structure.

A Process Documentation Record Schema

Below we give the full schema for process documentation record and record acknowledgement messages.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.pasoa.org/schemas/version023s1/record/PRecord.xsd"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified"
           xmlns:pr="http://www.pasoa.org/schemas/version023s1/record/PRecord.xsd"
           xmlns:ps="http://www.pasoa.org/schemas/version023s1/PStruct.xsd"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://www.pasoa.org/schemas/version023s1/PStruct.xsd ..\PStruct.xsd ">

    <xs:annotation>
     <xs:documentation>
     The Provenance Store Record schema
Author: Paul Groth Last
     Modified: 30 August 2005

        Copyright (c) 2006 University of Southampton
        See pasoalicense.txt for license information.
        http://www.opensource.org/licenses/mit-license.php

     </xs:documentation>
    </xs:annotation>


    <xs:import
     namespace="http://www.pasoa.org/schemas/version023s1/PStruct.xsd"
     schemaLocation="../PStruct.xsd" />

    <xs:element name="record" type="pr:Record" />

    <xs:element name="recordAck" type="pr:RecordAck"/>

    <xs:element name="content" type="pr:Content"  />

    <xs:complexType name="RecordAck">
        <xs:sequence>
            <xs:element name="synch_ack" minOccurs="0"
             maxOccurs="unbounded" type="pr:SynchAck">
             <xs:annotation>
             <xs:documentation>
             If the provenance store is being accessed under
             a synchronous connection, (i.e. remote procedure
             call style) the provenance store may return a
             synch_ack instead of an ack element. Under such
             a situation, the provenance store does not need
             to parse the incoming message in order to send
             an acknowledgment.
             </xs:documentation>
             </xs:annotation>
            </xs:element>
            <xs:element name="ack" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="contentName">
                            <xs:simpleType>
                                <xs:restriction base="xs:string">
                                    <xs:enumeration value="interactionPAssertion"/>
                                    <xs:enumeration value="actorStatePAssertion"/>
                                    <xs:enumeration value="relationshipPAssertion"/>
                                    <xs:enumeration value="exposedInteractionMetaData"/>
                                    <xs:enumeration value="submissionFinished"/>
                                </xs:restriction>
                            </xs:simpleType>
                        </xs:element>

                        <xs:element ref="ps:interactionKey"/>
                        <xs:element ref="ps:viewKind"/>
                        <xs:element ref="ps:localPAssertionId" minOccurs="0"/>

                    </xs:sequence>
                </xs:complexType>
            </xs:element>

            <xs:element name="ERROR" minOccurs="0" maxOccurs="1" type="xs:string">
             <xs:annotation>
             <xs:documentation>
             This field should be pressent if the messageName
             element's value is ERROR. As of yet we do not
             define the type of the error message that can be
             returned by the provenance store.
             </xs:documentation>
             </xs:annotation>
            </xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="Record">
        <xs:sequence>
        <xs:element name="identifiedContent" type="pr:IdentifiedContent" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="IdentifiedContent">
     <xs:sequence>
     <xs:element ref="ps:interactionKey"/>
     <xs:element ref="ps:viewKind"/>
     <xs:element ref="ps:asserter"/>
     <xs:element ref="pr:content" maxOccurs="unbounded"/>
     </xs:sequence>
    </xs:complexType>


    <xs:complexType name="Content">
     <xs:choice>
     <xs:element ref="ps:interactionPAssertion"/>
     <xs:element ref="ps:actorStatePAssertion" />
     <xs:element ref="ps:relationshipPAssertion"/>
     <xs:element ref="ps:exposedInteractionMetaData"/>
     <xs:element name="submissionFinished" type="xs:int" />
     </xs:choice>
    </xs:complexType>

    <xs:complexType name="SynchAck"></xs:complexType>

</xs:schema>
```

B Process Documentation Record WSDL

Below we give the WSDL document for process documentation record and acknowledgement messages.

```
<?xml version="1.0"?>

<definitions name="PRecord"
                   targetNamespace="http://www.pasoa.org/schemas/version023s1/record/PRecord.wsdl"
                   xmlns:tns="http://www.pasoa.org/schemas/version023s1/record/PRecord.wsdl"
                   xmlns="http://schemas.xmlsoap.org/wsdl/"
                   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl"
                   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                   xmlns:pr="http://www.pasoa.org/schemas/version023s1/record/PRecord.xsd">

    <documentation>
        The Provenance Store recording port type and messages
        Author: Paul Groth
        Last Modified: Feb 1 2005

        Copyright (c) 2006 University of Southampton
        See pasoalicense.txt for license information.
        http://www.opensource.org/licenses/mit-license.php
    </documentation>

    <import namespace="http://www.pasoa.org/schemas/version023s1/record/PRecord.xsd"
      location="./PRecord.xsd" />


    <message name="Record">
        <part name="body" element="pr:record"/>
    </message>

    <message name="RecordAck">
        <part name="body" element="pr:recordAck"/>
    </message>

    <portType name="RecordPortType">
        <operation name="Record">
           <input message="tns:Record"/>
           <output message="tns:RecordAck"/>
        </operation>
    </portType>

</definitions>
```

References

[Bra97] Scott Bradner. Key words for use in RFCs to indicate requirement levels. http://www.ietf.org/rfc/rfc2119.txt, 1997.

[GJ+06] Paul Groth, Sheng Jiang, , Simon Miles, Steve Munroe, Victor Tan, Sofia Tsasakou, and Luc Moreau. An Architecture for Provenance Systems. Technical report, Electronics and Computer Science, University of Southampton, 2006.

[GLM04] Paul Groth, Michael Luck, and Luc Moreau. A protocol for recording provenance in service-oriented grids. In Teruo Higashino, editor, *Proceedings of the 8th International Conference on Principles of Distributed Systems (OPODIS'04)*, volume Lecture Notes in Computer Science, pages 124–139, Grenoble, France, December 2004. Springer-Verlag.

[MGJ$^+$06] Steve Munroe, Paul Groth, Sheng Jiang, Simon Miles, Victor Tan, and Luc Moreau. Data model for Process Documentation. Technical report, University of Southampton, June 2006.

[TGJ$^+$06] Victor Tan, Paul Groth, Sheng Jiang, Simon Miles, Steve Munroe, and Luc Moreau. WS Provenance Glossary. Technical report, Electronics and Computer Science, University of Southampton, 2006.

[TMG$^+$06] Victor Tan, Steve Munroe, Paul Groth, Sheng Jiang, Simon Miles, and Luc Moreau. The Provenance Standardisation Vision. Technical report, University of Southampton, June 2006.

[Var04] Various authors. Web Services Security. http://www-128.ibm.com/developerworks/library/specification/ws-secure/, 2004.

[Var05a] Various authors. Web Services Secure Conversation Language. http://www-128.ibm.com/developerworks/webservices/library/specification/ws-secon/, 2005.

[Var05b] Various authors. Web Services Trust Language. http://www-128.ibm.com/developerworks/library/specification/ws-trust/, 2005.

[W3C99] W3C. XML Path Language (XPath) Version 1.0. W3C Recommendation 16 November 1999. http://www.w3.org/TR/xpath, 1999.