ws-prov-pquery

Authors:
Simon Miles, U. Southampton
Luc Moreau, U. Southampton
Paul Groth, U. Southampton
Victor Tan, U. Southampton
Steve Munroe, U. Southampton
Sheng Jiang, U. Southampton

November 23, 2006

Provenance Query Protocol

Status of this Memo

This document provides information to the community regarding the specification of a protocol for querying the provenance of data items from process documentation and has the status of a working draft. It does not define any standards or technical recommendations. Distribution is unlimited.

Copyright Notice

Abstract

A related document [MGJ+06] defines schemas to be used for documentation about the execution of a process, *process documentation*. It also defines the provenance of a data item as the process that led to that item. A *provenance query* is a query for the provenance of a data item and the results of such a query is documentation of the process that led to the item. In this document, we specify a protocol by which a querying actor and provenance store can communicate in performing a provenance query. This primarily takes the form of an abstract WSDL interface defining messages to be accepted and produced by a provenance store.

Contents

1 Introduction

The amount of information making up the provenance of an entity may be vast. The details of everything that ultimately caused an entity to be as it is would, generally, be an unmanageable amount. For example, to give the full provenance of an experiment's results, we have to describe the process that produced the materials used in the experiment, the provenance of materials used in producing those materials, the devices and software used in the experiment and their settings, the origin of the experiment design etc. Ultimately, if enough information was available, we would include details of processes back to the beginning of time. Similarly, given enough information, we could give finer and finer grained information on the processes that led to an entity, e.g. not just documenting that a sample was tested to see if a chemical was present, but the procedure by which this is done, the molecular interactions that took place in the testing procedure and so on. All the information about the provenance of an entity is potentially useful for someone with a particular question about that entity, but providing it all in all cases would be counter-productive.

Instead, anyone requiring the provenance of an entity should be able to get it by expressing the request as a *query* and *scoping* that query so that only the information relevant to them is returned. We define a *provenance query engine* as the actor that processes a provenance query and returns the results.

This document defines the schema for a provenance query request, the behaviour expected in processing that request and the resulting response. The WSDL 1.1 description of an interface for a service accepting such requests and producing such responses is given in Appendix B.

1.1 Goals and Requirements

The goal of this document is to define the protocol for querying the provenance of data items from process documentation.

1.1.1 Requirements

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

- Define the schema of the provenance query request message sent to the provenance query engine. This definition includes how the data item of which to find the provenance is identified, and how the scope of the query results are expressed.
- Define the behaviour of a provenance query engine on receiving a provenance query request.
- Define how provenance queries over distributed process documentation can be resolved using the links defined in a supporting document [MTG+06].

- Define the schema of the provenance query results message returned by the provenance query engine.

1.1.2 Non-Requirements

This document specifies a synchronous version of the query protocol. Other documents may specify asynchronous provenance querying.

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/xquery/XQuery.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)
pq	`http://www.pasoa.org/schemas/version023s1/pquery/ProvenanceQuery.xsd`	[PQuery]
ps	`http://www.pasoa.org/schemas/version023s1/PStruct.xsd`	[PStruct]
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 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

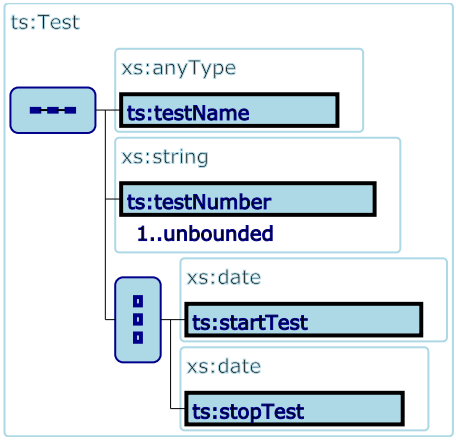

Figure 1: An example XML Schema diagram

`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 Provenance Query

To execute a provenance query, a *provenance query request* is sent to a *provenance query engine* by a *querying actor* and a *provenance query result* is returned. We specify below the request document schema, provenance query engine behaviour and response document schema for a provenance query. The full schema document, in which request and response 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.

4 Provenance Query Request

A provenance query request is a message sent by a querying actor to a provenance query engine to perform a query to find the provenance of a data item over the contents of that, and linked, stores. A provenance query request includes a *query data handle*, identifying the entity of which to find the provenance, and a *relationship target filter*, specifying the query's scope.

A provenance query request is instantiated as a document conforming to the schema depicted in Figure 2.

`/pq:provenanceQuery`

This element defines a provenance query request.

`/pq:provenanceQuery/pq:queryDataHandle`

This element defines a search over process documentation to find the record of an entity at a given event for which the querying actor wishes to find the provenance.

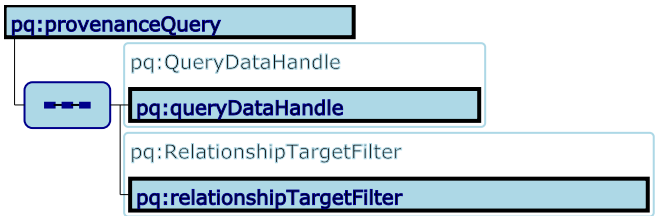

Figure 2: Provenance Query Request

`/pq:provenanceQuery/pq:relationshipTargetFilter`

This element contains the criteria by which the querying actor specifies whether any given entity in the documentation, and its relations, should be included in the query results.

4.1 Query Data Handle

When a querying actor asks for an entity's provenance, it identifies the entity such that a provenance query engine can find documentation of the entity. The identification is called a *query data handle*. For the actor, a query data handle identifies an entity at a given event. For the engine, a query data handle identifies a search for p-assertion data items in process documentation.

Not all p-assertion contents may be in the format required by the search expression. In this case, the querying actor can provide a set of *document language mappings*, which specify how to convert formats so that the search can take place. 0 mappings mean that all the p-assertion contents are expected to be in the language required by the search, e.g. XML. More than one mapping means that different p-assertion contents are mapped, e.g. one for CORBA messages, one for BLAST results etc.

A query data handle is instantiated as a document conforming to the schema depicted in Figure 3.

`/pq:queryDataHandle`

This element contains all details used to discover documentation of a data item.

`/pq:queryDataHandle/pq:search`

This element includes a definition of a search over process documentation to find the record of an entity at a given event for which the querying actor wishes to find the provenance. Different provenance query engines may support different types of search specification.

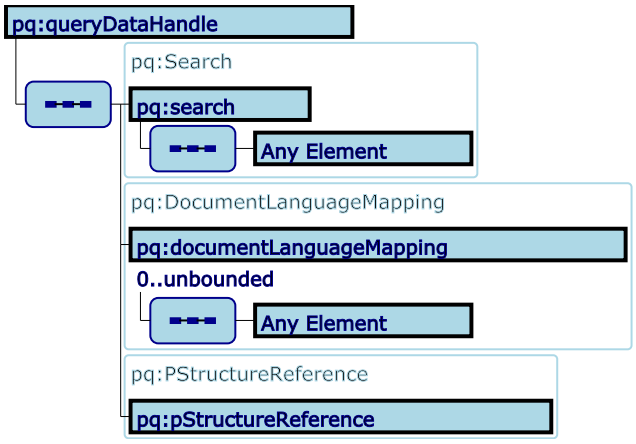

Figure 3: Query Data Handle

`/pq:queryDataHandle/pq:documentLanguageMapping`

This element includes a specification of how p-assertion contents are mapped to the document language required by the search, e.g. if p-assertion contents are comma-separated values, it could be mapped to an XML format for the search. The mappings required depend on the search language, and so different provenance query engines may support different types of search specification.

`/pq:queryDataHandle/pq:pStructureReference`

This element includes a definition of the exact set of process documentation over which the query data handle searches.

A `pq:pStructureReference`, which identifies the set of process documentation over which the query data handle search will be performed, is instantiated as a document conforming to the schema depicted in Figure 4.

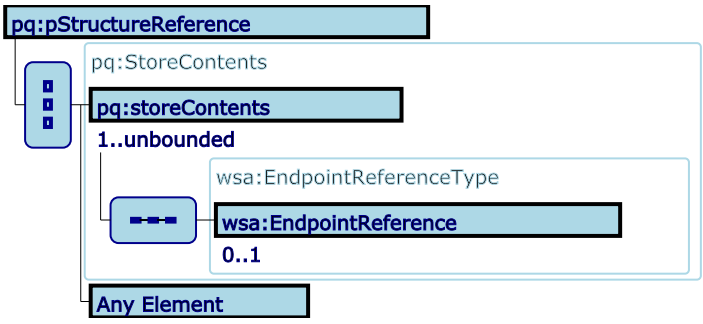

Figure 4: P-Structure Reference

`/pq:pStructureReference`

This element includes a definition of the exact set of process documentation over which the query data handle searches.

`/pq:pStructureReference/pq:storeContents`

The presence of this element is a statement that the query data handle search should be performed over the full contents of a provenance store. This element may occur multiple times, to specify that the search is over multiple stores. Where the element has no contents, this states that the search should be performed over the default store for the provenance query engine, usually the store that the engine is deployed as a component of.

`/pq:pStructureReference/pq:storeContents/wsa:EndpointReference`

If present, this element gives the address of a provenance store over which the search is conducted. This is a provenance store service port as defined in [MTG+06].

`/pq:pStructureReference/xs:any`

Where the storeContents element is not used, the set of process documentation over which the query data handle searches must be specified in another way. This element, if present, defines this search space in a way that the provenance query engine can interpret.

4.2 P-Assertion Data Key Query Data Handles

All provenance query engines can interpret query data handles of a particular form: a p-assertion data key as specified in the process documentation data model. This search will find at most one p-assertion data item, i.e. the one referred to by the key.

An example is shown below.

```
<pq:queryDataHandle>
  <pq:search>
    <ps:pAssertionDataKey>
      <ps:interactionKey>
        <ps:messageSource>...</ps:messageSource>
        <ps:messageSink>...</ps:messageSink>
        <ps:interactionId>...</ps:interactionId>
      </ps:interactionKey>
      <ps:viewKind xsi:type="ps:SenderViewKind"/>
      <ps:localPAssertionId>...</ps:localPAssertionId>
      <ps:dataAccessor>...</ps:dataAccessor>
    </ps:pAssertionDataKey>
  </pq:search>
  <pq:pStructureReference>
    <pq:storeContents/>
  </pq:pStructureReference>
<pq:queryDataHandle>
```

4.3 Relationship Target

A p-assertion data item is a part, or whole, of a p-assertion's content. A *relationship target* is the full set of information about a p-assertion data item that is the subject or object of a relationship p-assertion. It is used to determine whether a p-assertion data item is within scope for the provenance query results or not.

A relationship target is instantiated as a document conforming to the schema depicted in Figure 5.

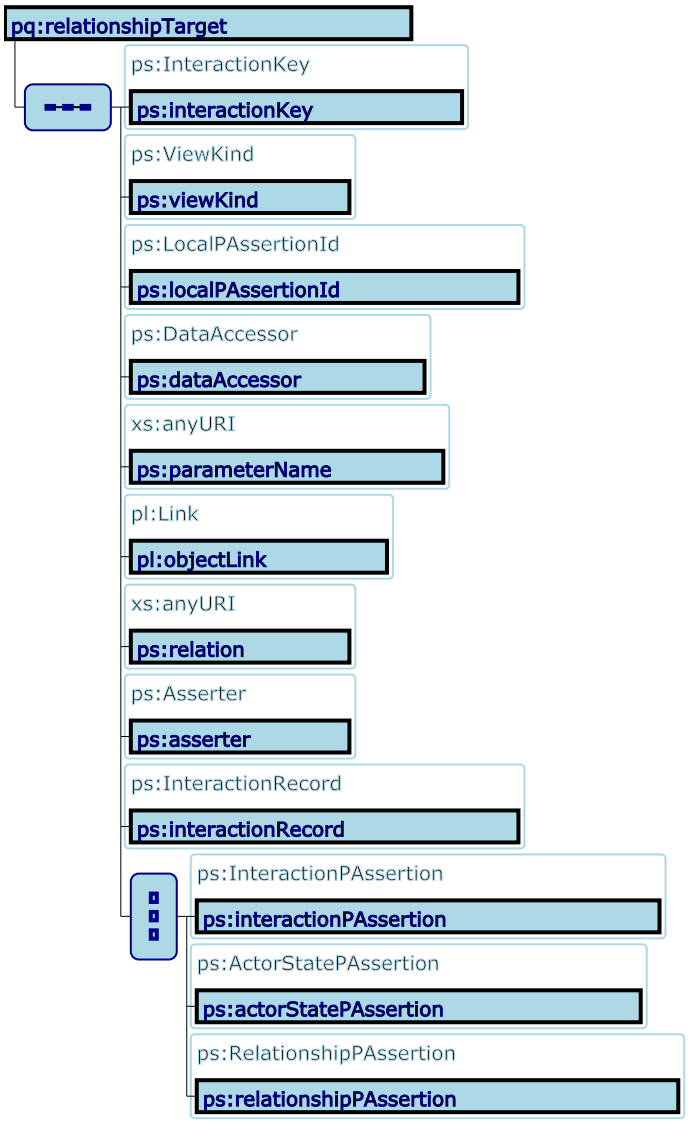

Figure 5: Relationship Target

`/pq:relationshipTarget`

This element contains all the information on one p-assertion data item.

`/pq:relationshipTarget/ps:interactonKey`

This is the interaction key of the interaction that the p-assertion containing the data item documents.

`/pq:relationshipTarget/ps:viewKind`

This is the view kind of the asserter of the p-assertion containing the data item.

`/pq:relationshipTarget/ps:localPAssertionId`

This is the local p-assertion ID of the p-assertion containing the data item.

`/pq:relationshipTarget/ps:dataAccessor`

This is the location of the data item within the content of its containing p-assertion.

`/pq:relationshipTarget/ps:parameterName`

This is the role played by the data item in the relationship of which this item is a target.

`/pq:relationshipTarget/pl:objectLink`

This is a link to the provenance store in which the p-assertion data item is documented.

`/pq:relationshipTarget/ps:relation`

This is the type of the relationship of which the data item is a target.

`/pq:relationshipTarget/ps:asserter`

This is the identity of the asserter of the p-assertion containing the data item.

`/pq:relationshipTarget/ps:interactionRecord`

This is the full set of p-assertions recorded about the interaction in which the data item is asserted.

`/pq:relationshipTarget/ps:interactionPAssertion`

If present, this is the interaction p-assertion in which the data item is contained.

`/pq:relationshipTarget/ps:actorStatePAssertion`

If present, this is the actor state p-assertion in which the data item is contained.

`/pq:relationshipTarget/ps:relationshipPAssertion`

If present, this is the relationship p-assertion which is itself the object of a relationship (and so the data item in this context).

4.4 Relationship Target Filter

The set of process documentation that may be returned in response to a provenance query request could be vast, and most of it irrelevant to a querying actor for any one purpose. Therefore, we need to allow the querying actor to specify the scope of the provenance query, i.e. a definition of what documentation is relevant enough to be part of the query results. This is the purpose of the *relationship target filter*. A relationship target filter is used to filter each object of a relationship p-assertion found while evaluating the provenance query, and filters out those that are not in scope. To do this, the provenance query engine constructs a relationship target for each relationship object, which encapsulates all relevant data about the data item the object identifies. The relationship target filter then defines a function that operates over a relationship target and returns true or false depending on whether the relationship target is in scope or not.

A relationship target filter is instantiated as a document conforming to the schema depicted in Figure 6.

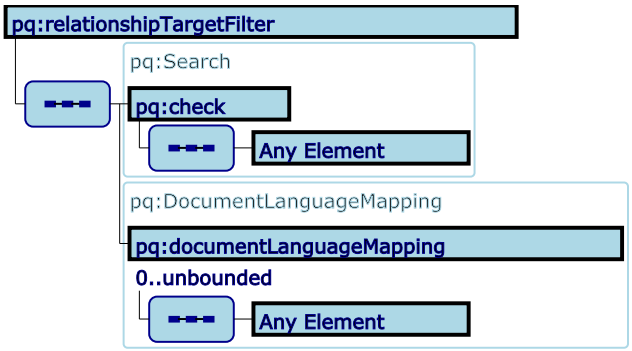

Figure 6: Relationship Target Filter

`/pq:relationshipTargetFilter`

This element includes a specification of the scope of a provenance query.

`/pq:relationshipTargetFilter/pq:check`

This element contains a definition for a function that operates on a relationship target that determines whether the target is in scope or not.

`/pq:relationshipTargetFilter/pq:check/xs:any`

This is the function definition. The form depends on what is supported by the provenance query engine, and may be suited to a particular type of process documentation.

`/pq:relationshipTargetFilter/pq:documentLanguageMapping`

This element includes a specification of how p-assertion contents are mapped to the document language required by the filter, e.g. if p-assertion contents are comma-separated values, it could be mapped to an XML format for the search. The mappings required depend on the search language, and so different provenance query engines may support different types of search specification.

5 Behaviour

A provenance query request is processed as follows.

1. The search for a data item expressed by the query data handle is performed on the process documentation contained in the given search space (`pStructureReference`). The result of this search is a set of p-assertion data items, addressed by p-assertion data keys.

2. For each relationship, asserted in the process documentation, of which one of those p-assertion data items is a subject, take each object of the relationship, express it as a relationship target, and execute the relationship target filter on it. As the provenance store containing the data item may not include both views of the interaction in which the data item was exchanged, the other view, referred to by a view link, may be retrieved by submitting process documentation and provenance queries to the linked store.

3. Where a relationship object is accepted by the relationship target filter, find the provenance of the p-assertion data item referred to by that object using the steps above and the same relationship target filter. As the object may be in another provenance store, specified by the object link, this may involve submitting process documentation and provenance queries to that store.

4. The final results of the query are comprised of two parts: the p-assertion data keys for every item discovered by the query data handle search; and, for every relationship object accepted by the relationship target filter, the relationship between that object and the subject of the relationship.

5.1 Data Accessors

The process documentation data model allows data accessors of any form to be used in the subjects and objects of relationship p-assertions. It is left to querying actors to correctly interpret these. However, a provenance query engine, following the above algorithm, must also be able to interpret them in order to

determine whether the object of one relationship is the same as the object of another relationship. The exact data accessors that a provenance query engine can understand is not restricted, but in order to use them the engine must be able to perform the following operations. For each operation, we state exactly why it is needed, i.e. what necessary operation would be impossible if the operation was not defined.

Get Accessor For Item Get the data accessor for a result of a query data handle search. Without this operation, an engine is unable to determine whether the subject of a relationship p-assertion refers to a data item found by the query data handle.

Test Accessor Equality Given the data accessor in the subject of a relationship and the data accessor in an object of a relationship, determine whether they refer to the same item. This may or may not involve navigating the p-assertion content. Without this operation, an engine cannot iteratively follow the relationships of each data item in the process.

6 Provenance Query Result

The final results of the query are comprised of two parts: the p-assertion data items from the query data handle search (the start data items); and, for every relationship object accepted by the filter, the relationship between that object and the subject in that relationship.

A provenance query result is instantiated as a document conforming to the schema depicted in Figure 7.

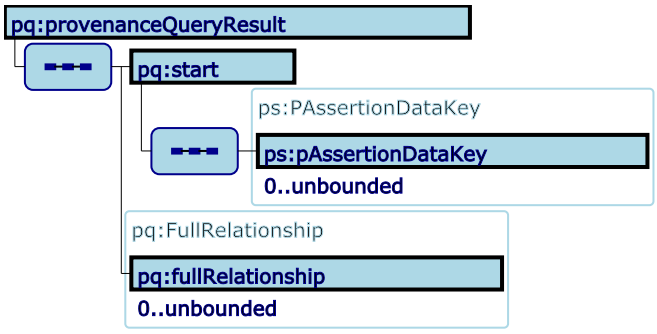

Figure 7: Provenance Query Result

`/pq:provenanceQueryResult/pq:start`

This element contains the p-assertion data keys for the items found by the query data handle search.

`/pq:provenanceQueryResult/pq:start/ps:pAssertionDataKey`

This is a p-assertion data key for an item found by the query data handle search.

`/pq:provenanceQueryResult/pq:fullRelationship`

This element contains the details of a relationship between two data items in the results of the provenance query.

6.1 Full Relationship

A *full relationship* is the full details of a relationship between two p-assertion data items. The information it contains is derived from a relationship p-assertion, but there are two differences. First, there is only a single object per full relationship, because each relationship object has been evaluated independently as to whether it is in scope for the provenance. Second, it is not contained within an interaction record, so the interaction key and view kind of the subject is included in a full relationship.

By matching a start p-assertion data key with those in the subjects and objects of full relationships, a graph of relationships describing the provenance of that start item can be determined.

A full relationship is instantiated as a document conforming to the schema depicted in Figure 8.

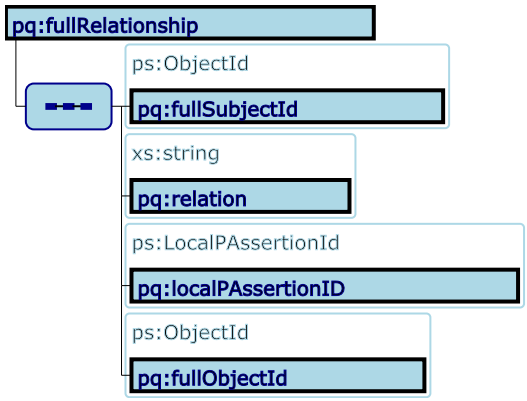

Figure 8: Full Relationship

`/pq:fullRelationship`

This element contains the details of a relationship between two p-assertion data items.

`/pq:fullRelationship/pq:fullSubjectId`

This is the subject of the relationship. It has exactly the same contents as the object ID of a relationship p-assertion as defined for the process documentation data model [MGJ+06].

`/pq:fullRelationship/ps:relation`

This is the type of relationship between subject and object.

`/pq:fullRelationship/ps:localPAssertionId`

This is the local p-assertion ID of the relationship p-assertion from which the full relationship is derived. The other identifiers of the p-assertion, its interaction key and view kind, are the same as in the fullSubjectId.

`/pq:fullRelationship/pq:fullObjectId`

This is the object of the relationship. It has exactly the same contents as the object ID of a relationship p-assertion as defined for the process documentation data model [MGJ+06].

6.2 Faults

Provenance query faults are exchanged by using extensions of the ProvenanceQueryFault type. A default element is specified for this type, and is shown in Figure 9.

Figure 9: Provenance Query Fault

`/pq:provenanceQueryFault`

This element, or extensions of it, represents a fault caused by the evaluation of a provenance query request.

7 Default Port Name

In order to aid addressing of provenance stores, we require that each interface that can be assumed to be present in every provenance store be given a default port context name [MTG+06], i.e. the last part of the URL addressing the port supporting that interface. For the provenance query interface, the default port name is `pquery`.

8 Security Considerations

This specification defines the process documentation query request and response messages for any provenance store supporting the process documentation query interface. 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.

8.1 Securing Message Exchanges

When messages are exchanged between a querier and provenance store in a process documentation query, 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 (query expression or results) 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 querier frequently performs process documentation queries on a store 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.

8.2 Securing Provenance Store Contents

Since this specification defines a mechanism to retrieve the contents of provenance stores, security policies should be established that ensure that only authorized queriers can access the p-assertions.

9 Conclusions

The aim of a process documentation data model is, first among many requirements, to enable the provenance of items to be determined. In this document, we have presented the data models for expressing provenance queries and their results, and stated the behaviour that can be expected of an engine evaluating such a query.

A Provenance Query Schema

Below we give the full schema for provenance queries and their results.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.pasoa.org/schemas/version023s1/pquery/ProvenanceQuery.xsd"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:ps="http://www.pasoa.org/schemas/version023s1/PStruct.xsd"
           xmlns:pq="http://www.pasoa.org/schemas/version023s1/pquery/ProvenanceQuery.xsd"
           xmlns:pl="http://www.pasoa.org/schemas/version023s1/distribution/PLinks.xsd"
           xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">

      <xs:annotation>
    <xs:documentation>
         Author: Simon Miles
      Last Modified: 28 Feb 2006

                 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:import namespace="http://www.pasoa.org/schemas/version023s1/distribution/PLinks.xsd"
                schemaLocation="../distribution/PLinks.xsd"/>
     <xs:import namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing"
                schemaLocation="../wsaddressing.xsd"/>

  <xs:element name="provenanceQuery" type="pq:ProvenanceQuery"/>

  <xs:element name="provenanceQueryResult" type="pq:ProvenanceQueryResult"/>

  <xs:element name="provenanceQueryFault" type="pq:ProvenanceQueryFault"/>

  <xs:complexType name="ProvenanceQueryFault"/>

  <xs:complexType name="ProvenanceQuery">
    <xs:sequence>
      <xs:element ref="pq:queryDataHandle"/>
      <xs:element ref="pq:relationshipTargetFilter"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="ProvenanceQueryResult">
    <xs:sequence>
      <xs:element name="start">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="ps:pAssertionDataKey" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element ref="pq:fullRelationship" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:element name="queryDataHandle" type="pq:QueryDataHandle"/>
```

```
<xs:element name="documentLanguageMapping" type="pq:DocumentLanguageMapping"/>

<xs:complexType name="DocumentLanguageMapping">
  <xs:sequence>
    <xs:any/>
  </xs:sequence>
</xs:complexType>

<xs:element name="pStructureReference" type="pq:PStructureReference"/>

<xs:element name="storeContents" type="pq:StoreContents"/>

<xs:element name="relationshipTargetFilter" type="pq:RelationshipTargetFilter"/>

<xs:complexType name="PStructureReference">
  <xs:choice>
    <xs:sequence>
      <xs:element ref="pq:storeContents" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:any/>
  </xs:choice>
</xs:complexType>

<xs:complexType name="StoreContents">
  <xs:sequence>
    <xs:element ref="wsa:EndpointReference" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="QueryDataHandle">
  <xs:sequence>
    <xs:element name="search" type="pq:Search"/>
    <xs:element ref="pq:documentLanguageMapping" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="pq:pStructureReference"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="RelationshipTargetFilter">
  <xs:sequence>
    <xs:element name="check" type="pq:Search"/>
    <xs:element ref="pq:documentLanguageMapping" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="relationshipTarget" type="pq:RelationshipTarget"/>

<xs:complexType name="RelationshipTarget">
  <xs:sequence>
    <xs:element ref="ps:interactionKey"/>
    <xs:element ref="ps:viewKind"/>
    <xs:element ref="ps:localPAssertionId"/>
    <xs:element ref="ps:dataAccessor"/>
    <xs:element ref="ps:parameterName"/>
    <xs:element ref="pl:objectLink"/>
    <xs:element ref="ps:relation"/>
    <xs:element ref="ps:asserter"/>
    <xs:element ref="ps:interactionRecord"/>
    <xs:choice>
      <xs:element ref="ps:interactionPAssertion"/>
      <xs:element ref="ps:actorStatePAssertion"/>
      <xs:element ref="ps:relationshipPAssertion"/>
    </xs:choice>
  </xs:sequence>
</xs:complexType>
```

```
  <xs:element name="fullSubjectId" type="ps:ObjectId"/>

  <xs:element name="fullObjectId" type="ps:ObjectId"/>

  <xs:complexType name="FullRelationship">
    <xs:sequence>
      <xs:element ref="pq:fullSubjectId"/>
      <xs:element ref="ps:relation"/>
      <xs:element ref="ps:localPAssertionID"/>
      <xs:element ref="pq:fullObjectId"/>
    </xs:sequence>
  </xs:complexType>

  <xs:element name="fullRelationship" type="pq:FullRelationship"/>

  <xs:complexType name="Search">
    <xs:sequence>
      <xs:any/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

B Provenance Query WSDL

Below we give the WSDL document for provenance query and response messages.

```
<?xml version="1.0"?>
<definitions name="PQuery"
             targetNamespace="http://www.pasoa.org/schemas/version023s1/pquery/PQuery.wsdl"
             xmlns:tns="http://www.pasoa.org/schemas/version023s1/pquery/PQuery.wsdl"
             xmlns:pq="http://www.pasoa.org/schemas/version023s1/pquery/ProvenanceQuery.xsd"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns="http://schemas.xmlsoap.org/wsdl/">

    <documentation>
        The Provenance Store provenance query port type and messages
        Author: Simon Miles
        Last Modified: 28 Feb 2006

        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/pquery/ProvenanceQuery.xsd"
            location  = "./ProvenanceQuery.xsd"/>

<!-- Defines the Provenance Query Port type which offers one operation:
           the ProvenanceQuery operation. This operation takes in a ProvenanceQuery message defined
           by the pq:provenanceQuery element in ProvenanceQuery.xsd. The operation returns an output
           message ProvenanceQueryResult defined by the pq:provenanceQueryResult element in ProvenanceQuery.xsd  -->

    <portType name = "PQueryPortType">
        <operation name = "ProvenanceQuery">
           <input message = "tns:ProvenanceQuery"/>
           <output message = "tns:ProvenanceQueryResult"/>
           <fault message = "tns:ProvenanceQueryFault"/>
        </operation>
    </portType>

    <message name = "ProvenanceQuery">
        <part name = "body" element = "pq:provenanceQuery"/>
    </message>

    <message name = "ProvenanceQueryResult">
        <part name = "body" element = "pq:provenanceQueryResult"/>
    </message>

    <message name = "ProvenanceQueryFault">
        <part name = "body" element = "pq:provenanceQueryFault"/>
    </message>
</definitions>
```

References

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

[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.

[MTG+06] Steve Munroe, Victor Tan, Paul Groth, Sheng Jiang, Simon Miles, and Luc Moreau. WSRF Data Model Profile for Distributed Provenance. 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.

[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.