UNIVERSITY OF SOUTHAMPTON

Provenance-based Data Traceability Model and Policy Enforcement Framework for Cloud Services

by

Mufajjul Ali

Supervisor: Prof Luc Moreau

A thesis submitted in partial fulfilment for the degree of Doctor of Engineering

in the

Faculty of Engineering and Applied Science

Department of Electronics and Computer Science

March 2016

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND APPLIED SCIENCE
DEPARTMENT OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Engineering

by Mufajjul Ali

In the context of software, provenance holds the key to retaining a reproduceable instance of the duration of a service, which can be replayed/reproduced from the beginning. This entails the nature of invocations that took place, how/where the data were created, modified, updated and the user's engagement with the service.

With the emergence of the cloud and the benefits it encompasses, there has been a rapid proliferation of services being developed and adopted by commercial businesses. However, these services expose very little internal workings to their customers, and insufficient means to check for the right working order. This can cause transparency and compliance issues, especially in the event of a fault or violation, customers and providers are left to point finger at each other.

Provenance-based traceability provides a means to address a part of this problem by being able to capture and query events that have occurred in the past to understand how and why it took place. On top of that, provenance-based policies are required to facilitate the validation and enforcement of business level requirements for end-users satisfaction.

This dissertation makes four contributions to the state of the art: i) By defining and implementing an enhanced provenance-based cloud traceability model (cProv), that extends the standardized Prov model to support characteristics related to cloud services. The model is then able to conceptualize the traceability of a running cloud service. ii) By the creation of a provenance-based policy language (cProvl) in order to facilitate the declaration and enforcement of the business level requirements. iii) By developing a traceability framework, that provides client and server-side stacks for integrating service-level traceability and policy-based enforcement of business rules. iv) Finally by the implementation and evaluation of the framework, that leverages on the standardized industry solutions. The framework is then applied to the commercial service: 'ConfidenShare' as a proof of concept.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to start by thanking God, the All Mighty for giving me this opportunity, the strength and courage to overcome many hurdles and challenges over the course of the EngD.

I am very humbled and honoured to have had the stupendous support, help and guidelines from my influential supervisor Prof. Luc Moreau. His determination to succeed, patience and perseverance through difficult times and thought provoking analysis on the research work were beyond exception and were the prominent drivers behind reaching my EngD's goals.

Being physically located in London and with the existing day-to-day commitments at Orange, balancing the University and industrial needs on occasions proved considerably challenging. However, such commitments were made manageable by the highly professionalism of the Southampton institution and organizational support of the managers at Orange.

No doubt, the tremendous backing of my line manager Kashif Chaudhry and the Technical Director Raffel Uddin were highly influential in initiating and retaining EngD's activities at Orange. It was a privilege to have had wonderful colleagues: Saiful Alom, Tansir Ahmed and Mobeen Ali for their encouragements, support, kind words, inspiring and motivating me through thick and thin, which is highly appreciated.

I would like to personally thank Abdellatif Benjelloun Touimi, for his valiant effort in laying the foundation for the EngD. It was his initiative that paved the way for the joint collaboration between Orange and Southampton University.

Last but not least, my prolonged EngD journey from beginning to the end would not have been sustainable without the unequivocal support from my family, especially my mother, missus and siblings. Many sacrifices and compromises were made to accommodate and prioritize the EngD's need. Such were their level of understanding and appreciation of the situation, it can only be described as a true blessing.

Chapter 1

Introduction

This chapter provides an overview of the industrial problem this dissertation addresses, and the contributions to the state of the art. It starts by introducing the Telco industry and their key area of interest.

1.1 The Telco Industry

Telecommunication operators have been very successful in offering traditional voice and SMS services to their customers over the past few years. With the advancement of 3G/4G technologies providing an ever-increasing data bandwidth, there has been a proliferation of services available on the devices, primarily from the OTT (over-the-top) players (such as Apple, Google, etc.) principally depending upon this increased bandwidth. While data revenues are on the increase, the traditional services are being adversely affected due to VOIP (voice over IP) and other messaging clients (such as Skype, WhatsApp and others). This decline in revenue affected by the Telcos is primarily due to the OTT players not having the overhead cost of running, managing or maintaining the telecommunication network. Therefore, they can solely focus on the service delivery at a much cheaper rate then the Telco operators themselves and in many cases, transport their services for free using the Telcos' signalling plane or control network layers. This is having a substantial negative impact on the Operators' revenue, where the bulk of their revenues are still from the traditional services.

The reality is that Telecom companies can no longer purely rely on the traditional services to maintain the operational, running and maintenance costs and thus generate and still retain a healthy profit margin. It is paramount for them to evolve, and seek other potential areas of revenue stream. This has led Orange (a multinational Telecom provider, with a customer base of more than 217m operating in more than 166 countries [166]) to explore other avenues such as cloud computing [167].

It was predicted by Gartner [155] that cloud spending was to reach $150bn by 2013 and $230bn (IHS [108]) by 2017. These figures now appear to be overly optimistic if we consider the IDC [201] projected cloud spending of 2015 which stands at 70bn (based on actual spending of 2014). However, these figures do indicate a continuous spending growth on a year-by-year basis and is likely to continue for the foreseeable future. The bulk of spending was forecast by IDC [21] to be on business applications at 52%, followed by infrastructure services at 22%. Interestingly, the share predicted for mobile cloud services was around $9.5bn by 2014 (see Figure 1.1). With the rise of smart phones and mobile devices adoption by users worldwide [199], [17], [160], we expect this figure to rise significantly over the next few years.

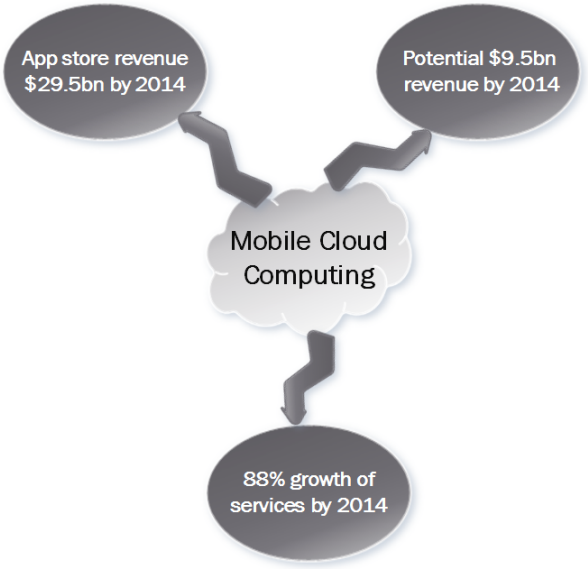


FIGURE 1.1: Mobile Cloud Services

The potential revenue to be generated from cloud computing/mobile cloud computing is very attractive to all operators including Orange (it would be in line with the interests of the group). However, there remains the issue of trust.

Trust has become the operative word in cloud computing [105]. Although trust of cloud services by Telcos is increasing [205], another form of trust remains one of the main concerns - that is with regards to *transparency* and *compliance* of data including its integrity over time [161], [212]. This has been echoed by many industry experts including David Mitchell Smith, a VP and Gartner fellow, who states 'The biggest pain point associated with security and management in the cloud is that of trust, ..., ***a policy or a strategy of trust-verify would be a prudent one***' [200]. In other words, having a mechanism that checks if a service is functioning as expected.

This concern is also shared by Orange, whereby the ability to demonstrate ***traceability (provenance) of a running cloud service*** *and satisfaction of* ***business level requirements through policy enforcement*** *are paramount to attracting, retaining and increasing loyalty of the customers.* This vision is championed by the churn manage-

ment of telecommunication strategy, where retaining customer loyalty is paramount for future growth.

One of the directions to address such concerns were through the EngD sponsorship. The motivation for the EngD can be seen from two perspectives: from a commercial and technological angle. The research findings should lead to some form of commercial benefit and give a competitive advantage over their rivals. At the same time, to make technological advancement that could bring an added dimension to the existing technologies in use today and enable a new breed of services that are currently not available in the market. The expectation is to apply the research findings to a commercial in-house cloud service by Orange called 'ConfidenShare', and evaluate it as a proof of concept. Orange is committed in providing the necessary operational and managerial supports for running of the EngD while technically being led by the University.

1.2 Traceability Enforcement in Cloud Services

Traceability is a fuzzy term used by various disciplines, although in most cases referring to a similar concept. For example, in the field of logistics, it is used to refer to the capability of tracing goods along the distribution chain. In measurement, it is used to refer to an unbroken chain of comparisons relating to an instrument's measurement. In forensic science, it is critical in investigating crimes leading to establish proofs against the perpetrators [106]. The idea of traceability in web-computing can be traced back in literature to as early as 1997. D.D. Steinauer *et al.* [203] sees the potential need for traceability to enhance trust in the E-Commerce sector, and is closely related to achieving accountability and monitoring for E-Commerce, running on dedicated servers, or through the cloud infrastructure platform.

Cloud computing relies on many existing tools and technologies in reducing the cost of service delivery whilst increasing the speed and agility of service deployment [215], supporting features such as the dynamic scaling, resource pooling, pay-per usage and on-demand self-services. The core technology behind cloud computing is virtualization [103], [211]; it empowers the whole cloud computing paradigm by creating an abstraction layer between the physical hardware and the operating system. This allows a greater degree of flexibility by being able to share the same physical resources virtually by more than one OS. Currently, there are three well defined service models: Infrastructure as a Service (IaaS [187]), Platform as a Service (PaaS) [54], and Software as a Service (SaaS)[104].

While cloud computing adoption is gaining momentum in industry, *the traceability remains its main Achilles heel* [4], [173]. In the virtualized environment, the actual physical data storage and computation can take place in one or more geographical locations around the world. The running instances of the virtual machines are selected primarily

based on the availability of hardware resources (in some cases with the option of region), which may not factor in or enforce any regulatory compliance or sensitive nature of the data it may be executing. This can raise legal and political issues since some jurisdictions may specifically require provision of this information. Data hosting in different countries might also be exposed to foreign untrusted entities/governments.

With the aid of virtualization, the underlying infrastructure for running a cloud service is completely abstract and hidden from the end-users. This allows the users to send and receive data without having to worry about the underlying details [153]. For normal users such level of abstraction may be satisfactory, but is inadequate for businesses/organizations where data may contain confidential or sensitive information. Knowledge of the underlying details of how the data is being shared, changed, modified and processed in the cloud would enable businesses to validate and verify of any breeches took place or check for compliance (rather than purely relying on their implicit trust of the cloud service provider). Such array of underlying details can be supported by having provenance of the cloud environment.

Provenance [217] is a well understood area in art and digital-libraries, where lineage, pedigree and source plays a major role in understanding how/where things have derived from, and in determining its authenticity and value [113]. In the cloud, provenance is fundamental in answering questions such as: What processes were involved in transforming the data? Who initiated the process? Did the processes conform to all necessary regulations? Where did the execution of data did take place (both from virtual to physical references)? Who had access to these data? In order to answer such questions, one needs to look at how provenance in the cloud can be modelled, captured and queried in the context of cloud computing. There has been a great deal of activity around defining provenance-based languages. Languages such as Proof Markup Language (PML) [176], Provenir ontology [191], Provenance Vocabulary, Dublin Core [23], WOT schema [90], SWAN provenance ontology [53], Semantic Web publishing Vocabulary [44] and Changeset vocabulary [210] have all been proposed. Interestingly, each of these languages, to a certain extent, can be mapped to the core of the Open Provenance Model (OPM) [146], [190]. OPM can be seen as the common subset of all these languages, however, it has been superseded by the W3C standardized model called Prov [148], [114] which provides extensibility to support domain specific requirements. Our approach to addressing the traceability issue in cloud service is through the use of provenance. In order to do this, we propose to extend the Prov by providing a domain specific model that is context specific to the cloud.

In addition to provenance, a policy enforcement mechanism is required to define the service requirements, and to act upon them (service requirements) if violation occurs. The current approach to addressing these is via access control. Access control (AC) plays a pivotal role in safeguarding systems from unauthorized access and providing different levels of access granularity. There has been several language specifications developed over

the years, ACL (Access Control List) [194], Role Based Access Control (RBAC)[73], Attribute Based Access Control (ABAC) [221], [204] and more recently policy based access control (PBAC) [223]. XACML [186], an industry wide standard, is deployed by many organizations for standard policy-based control. Its architecture is modular and provides scope for extensibility. However, it does not cater for handling of provenance data. A number of provenance-based policy languages that utilizes provenance data in their decision process of provenance information have been proposed [184], [42], [162] in recent years. While the degree of languages expressibility on the structure of provenance data, their integration with cloud-based services and mapping to industrial standards varies considerably, no one particular language appears to fulfil our service requirements (these are discussed in detail in Chapter 2). We propose to create a new provenance-based policy language to model all our service requirements and validate them against traceability data. On top of that a framework to support the policy enforcement and integration with cloud-based services.

We now present the thesis statement and the contributions to the state of the art, followed by the recent publications.

1.3 Thesis Statement & Contributions

Our solution to the problem of traceability and enforcement in the cloud can be summarized in the following thesis statement.

A combination of provenance-based traceability and a policy language is an effective way to enable and enforce restrictions in sharing and storing of user's data in the cloud.

The following contributions are made to the state of the art:

First Contribution - *A provenance-based cloud traceability model that extends the standardized Prov model to support characteristics related to cloud services.*

The model is able to trace the origin of a piece of data, evolutions/changes it has undergone, as well capture various relationships related to operation invocations, virtual to physical abstraction of resources, and others.

Second Contribution - *A provenance-based policy language that facilitates the declaration and enforcement of the business level requirements by querying on the generated traceability data.*

The syntax of the language for declaring abstract policies is primarily based on the Prov notation [147]. It is designed to allow policy declaration of service-level requirements.

Beyond declaring policies with predefined values, it facilitates the declaration of dynamic values through variable referencing. The values of which are determined at the runtime. The runtime system of the policy language also retains the provenance of its policy execution for auditing purpose.

Third Contribution - *A traceability framework that provides client and server-side stacks for integrating service-level traceability and policy-based enforcement of business rules for existing and new cloud services.*

This enables seamless integration of all necessary components by using high-level APIs that hide the underlying mechanism for encoding and storing traceability data, and for invoking the traceability-based enforcement of resource sharing and storage.

Fourth Contribution - *An implementation and evaluation of the framework that leverage on standardized industrial solutions for a greater degree of adaptability and compatibility with existing services and their deployment infrastructures.*

It leverages on standards such as XACML, and Prov and provides necessary mappings to port our solution (cProv and cProvl) to their existing open-sourced implementations. The evaluation shows a successful integration of the framework with the ConfidenShare service, that enables generation and storage of traceability data that is used by policy engine to perform policy enforcement. The addition of the framework (and its components) with the ConfidenShare service adds reasonable overhead in relative to the service offering, and has minimal impact on the overall usability of the service.

1.4 Other Contributions

Contribution Towards W3C Prov Implementation Report - *The EngD's work on data-traceability model for the cloud (cProv) that extends W3C Prov Model was included as part of the Prov implementation report.* The purpose of this document is to demonstrate that the features provided in Prov are implementable and interoperable.

http://www.w3.org/TR/2013/WD-prov-implementations-20130312/ [96]

1.5 Publications

The following publications have been produced based on this dissertation's work.

1. M.Ali and L.Moreau. "A provenance-based policy control framework for cloud services", In Bertram Ludscher and Beth Plale, editors, Provenance and Annotation of Data and Processes, volume 8628 of Lecture Notes in Computer Science, pages 127 – 138. Springer International Publishing, 2015.

2. M.Ali, L.Moreau, "A Provenance-Aware Policy Language (cProvl) and a Data Traceability Model (cProv) for the Cloud", 2013 IEEE Int. Conf. on Cloud and Green Computing (CGC) 2013, pp. 479-486, doi:10.1109/CGC.2013.81

1.6 Presentation Overview

Chapter Two - This chapter provides a detailed description of the "ConfidenShare" cloud service that is developed by the sponsor. The service is designed to allow sharing of data among multiple participants. This chapter also defines a number of requirements which are presented as non-structured policies that are addressed by this research.

Chapter Three - This chapter presents the literature review of the current state of the art. It explores existing provenance models, provenance for the cloud, and provenance-based policy languages that are related to our work. As a result of this chapter, we have identified the needs for a cloud specific traceability model, a policy language that can leverage on the traceability data, and a framework that can integrate the traceability model and policy language with cloud services.

Chapter Four - This chapter provides a cloud traceability model (cProv) that extends the W3C standard Prov. The model is designed to be more specific to context of the cloud, which can model the traceability of the "ConfidenShare" service.

Chapter Five - This chapter presents a provenance-based policy language (cProvl) that is designed to model the service requirements as policies (defined in Chapter Two) and to validate them using traceability data based on the cProv model (proposed in Chapter Four).

Chapter Six - This chapter details the traceability framework's architecture, implementation design and integration. It presents various stacks, APIs and mapping between technologies to support the integration of the traceability handling based on the cProv model, and provenance-based policy enforcement (cProvl) for the ConfidenShare service.

Chapter Seven - This chapter evaluates the implemented solution for integrating traceability and policy language with the ConfidenShare service; focusing primarily on the performance and additional overheads.

Chapter Eight - Conclusion & future work.

Chapter 2

ConfidenShare Service and Its Requirements

This chapter provides a detailed description of the commercial service "ConfidenShare" [168] by Orange [166] and its requirements. From these requirements, the policies have been derived and then listed for their implementation fulfillments.

The service for this research has been chosen carefully by the industrial sponsor (Orange) to meet their long-term strategy for cloud offerings: "ConfidenShare" [168], a cloud service currently being developed by Orange. It has gained a lot of interest within the group for the potential value proposition in respect to trust and security it promises to their customers, which in turn can be a source of future revenue. It is believed that with the aid of this research, the service can offer unique selling points (traceability and traceability-based policy enforcement) that can provide a competitive edge in the market.

2.1 Service Description

ConfidenShare is a cloud service, which has been in development over the past few years in the research domain of Orange. It is anticipated to go on trial in the near future.

The service is designed for the sharing of sensitive and non-sensitive information such as files, meeting data, and other data with users within the cloud environment. It uses Proxy re-encryption [20], a cryptographic technique that allows the sharing of all or part of user's data with one or more parties [193] securely.

ConfidenShare is interoperable with many existing cloud providers. Figure 2.1 shows an overview of the service design. Users would typically register with the ConfidenShare service (step 1). Once registered, they have the option of using their own cloud resource

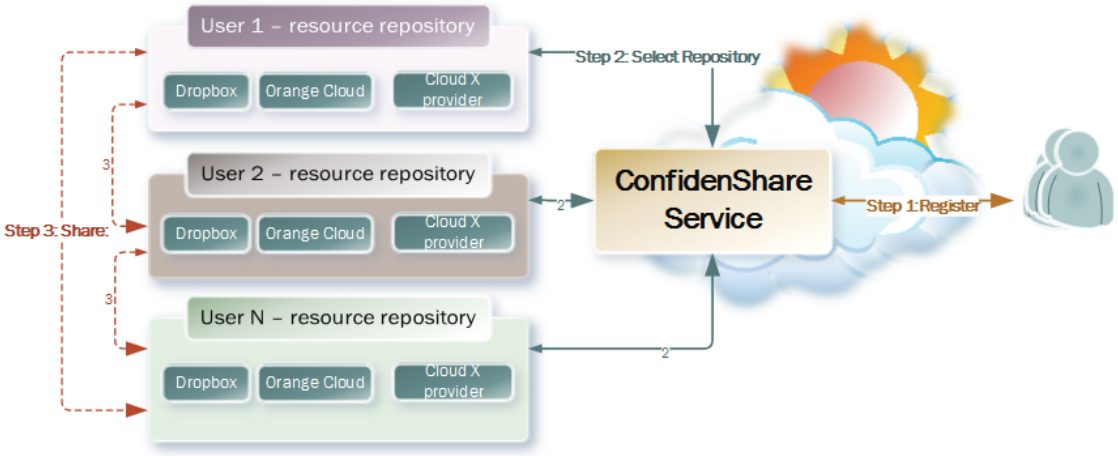


FIGURE 2.1: ConfidenShare Cloud Service

repository (such as Dropbox, One drive, and others) or the default repository provided by Orange (step 2). ConfidenShare would then mediate the storage and sharing of files between the repositories (step 3). Note, for implementation, we are only considering Orange cloud as the main source of repository since, we have full control over the underlying infrastructure.

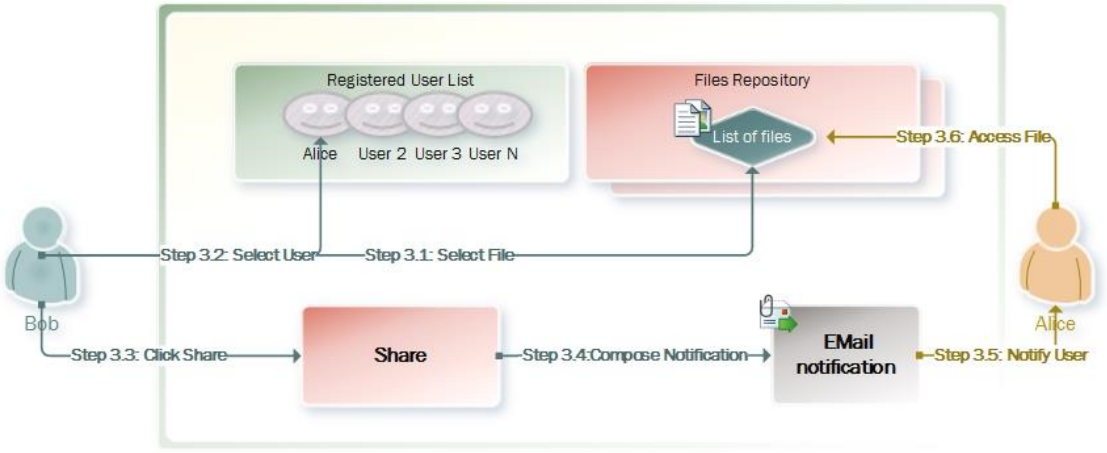


FIGURE 2.2: ConfidenShare - Share Operation

Figure 2.2 shows step 3 of Figure 2.1 in more details.

A user typically conducts a share operation by first selecting a file (step 3.1). If the file is not on the list, he/she would upload it to their chosen repository. The user would then select the recipient of the file to be shared with (step 3.2). He or she can choose to share the file by giving access to it as a link (URI) or an actual copy (replicated to the recipient's repository) (step 3.3). An email notification is composed with share details (step 3.4) and sent to the recipient. The recipient would follow the instructions to access the file from the sharers repository or from his/her repository (step 3.5).

2.2 Service Requirements

While the file sharing mechanism is secure, it does not have the necessary means of recording traceability, declaring constraints (service requirements) and performing enforcements to satisfy the constraints. This has led to the proposal of the following requirements in collaboration with the management, which are specific to this research.

There are various levels of confidentiality of sharing a file and the restrictions on the storage and its processing of it.

Files are typically categorized as 'confidential', 'restricted' or 'general', which are defined as follows.

A 'confidential' file is the most restrictive. Only the originator (creator of the document) is allowed to initiate the share. Re-sharing or modifications are not allowed by the recipients.

With a 'restricted' file, an originator can identify recipients and share restricted file with them. Recipients are allowed to modify and share between each other.

A 'general' file can be shared with any user, and there are no explicit restrictions on the re-sharing or modifying of it. This is the default category.

Only a registered user can share a file. For any user no longer registered with the service, the traceable files associated with that user cannot be shared by any of the recipients, and should be removed. This is in accordance to the "EU:Right to erasure" legislation [11]. If explicit permission has been given by the user (no-longer registered), the data can be retained and shared with other users.

Creation and upload of a 'confidential' file from a geographical location, requires the user to be logged on from the same geographical location (close proximity or within a defined boundary such as a region or a country) for the file to be shared. The 'restricted' and 'general' categories do not have such a restriction. A user can explicitly prevent the service from creating backups of the file by marking it as non-replicable during the creation of the file.

In order to validate the above requirements, historical operations performed by the service are required. The historical information (provenance data) would provide us with information such as what operation took place on the data, how the data were created, modified and changed. Such information can be used by the policy engine to check for location violations, file privileges & ownership restrictions and others.

The above descriptions are designed to give the user greater control of their data in respect to sharing, storing and processing of their data. For the purposes of this research, the requirements are presented as non-structured policies for their implementation that

require traceability data. These policies can be divided into three functions of the service: share, storage and processing.

2.2.1 Share Policies

Share requirements are an important aspect of the service that defines business logic on the handling and management of data when it is shared. *Please note the 'share' operation in these policies are considered to be* ***One-to-One****.* The share policies require traceability data to determine if all the constraints are satisfied for the share operation to take place. The policies are defined as follows:

Policy 1 - *A file (fileA) can only be shared by a registered user (userA) to another user (userB).*

Only authorized users are allowed to share a file. Non-registered users can only view the file, but will have to register in order to share.

Policy 2 - *If a file (fileA) is marked as 'confidential', only the originator is allowed to share it with another user (userB), re-sharing by the receiver (userB) to another user (userC) is not allowed.*

A confidential file typically contains highly sensitive data, and therefore it requires strict control to ensure unauthorized users do not have access to it.

Policy 3 - *If a file (fileA) is categorized as 'restricted', only the originator (userA) and the receivers (userB ... userN) are allowed to modify the file (fileA and its modifications fileA1 ... fileAn) and share (explicitly re-share) amongst themselves only.*

To determine if a re-share violation has occurred, traceability data is used to examine if any modifications took place in the file and if re-share is taking place among themselves. Note, currently, the re-share is manual. The ConfidenShare service does not have the support for automatic traceability recording of changes or modifications to the content of a file. However, certain file types (yet to be defined) may be supported in the future.

The restricted file data may contain confidential information, but it is not classified as sensitive.

Policy 4 - *If a file (fileA) is labelled as 'general', any user (UserA) can share the file with other users (UserX).*

These files are non-sensitive, and can be shared by anyone with everyone.

Policy 5 - *A user (userA) logged in from an authorised region (EU) cannot share a 'confidential' or 'restricted' file with another user (userB) logged in from an unauthorised region (non-EU).*

A ‘confidential‘ or ‘restricted ‘ file may contain sensitive information. Therefore, a user logged in from a different region, which may be subjected to regulatory restrictions or an un-trusted region, may pose a risk to the data. Files are classified as ‘general’ do not have such restrictions. Note, use of a secure VPN from an unauthorized region to an authorized region would be permitted since the VPN may reduce the overall security risk of being at the unauthorized region.

Policy 6 - *If a user (userA) is classed as ‘removed’, none of the associated shared files (file X) can be shared further.*

When a user is removed from using the service, by law, all the private data associated with the user must be deleted, this includes all the shared files. In cases where these shared files are not possible to be removed (due to share by copy), this policy can ensure that they cannot be shared further. The provenance of the deleted files remains, so that it can be used to check for the removal of the files.

2.2.2 Storage Policies

The storage requirements are designed to cater for the dynamic distributed nature of the cloud, and to ensure the data is handled appropriately. These requirements are also to be validated by traceability data, and can be defined as follows:

Policy 7 - *The storage of a new sensitive file (fileA) needs to reside in the same region as the registration of the user (userA), otherwise it is denied.*

For security reasons, a user’s file storage/creation needs to be located in close proximity to the registration region to ensure that the file is not exposed to unfavourable regulations/jurisdictions. Note, if a user is using their own third party repository, determining the storage location may be difficult, since the ConfidenShare service may not have direct access to such information, in such a case, it is recommend to use the Orange Cloud repository.

Policy 8 - *A system cannot replicate a file if the user explicitly marks it as non-replicable, at creation time.*

During the creation of a file, a user can choose if the file should be replicated or not. This policy enforces the explicit choice of the user and provides the user with cheaper service cost (due to less overhead cost), but this could potentially be risky - since no backups of the data are created.

2.2.3 Processing Policy

The processing requirements deal with the changes or modifications of the data to be in-line with the business needs. These are also modelled as policies and rely on traceability data for validating them.

Policy 9 - *A user (userA) can share a file (fileA), with non-modifiable preference, to another user (userB). This user (userB) cannot share a modified version of fileA (fileAA)*

This policy allows sharing of a file (such as a text file) that may not have been physicallly modified to read-only or non-modifiable, such as a PDF file. The traceability data in this case can be used to check if the file has been modified or not (assuming the permitted sotfware track changes). Note, the current check for modifications is manual. It is currently not supported by the ConfidenShare service, but will be considered for the future releases.

2.3 Service Requirements Satisfaction

The current service requirements clearly define a number of constraints that are needed to be fulfilled for the service to be commercially viable. While these constraints have been abstractly presented as policies, a concretely defined policy language with proper syntax and semantics is required to model and execute the policies.

The policies (see Section 2.2.1, 2.2.2, 2.2.3) are required to check for various forms of violations (location, file privileges, etc.). Such checks require traceability information associated with the historical operations (related to the sharing, storage and processing of data) carried out by the service to determine the right course of action(s) to be performed (e.g. permit, deny, etc.).

Policies related to sharing of data (2.2.1) can contain information with various level of restrictions (restricted, confidential and general) that are needed to be handled differently by the receiving user. In order to check if these restrictions are being adhere to, traceability data is required to be matched against.

Policies associated with storage of files (2.2.2) contain restrictions with regards to their storage and access locations. Traceability data retains information related to how the data files were stored and where it is being accessed from, with respect to their virtual and physical locations (with restriction details). Such information can be used by the policies to match for violations.

Policies containing processing restrictions (2.2.3) are required to check for changes of dynamic of the files. The provenance data possess information related what changes or processing occurred on the file. This information can be used to check for violations

related non-permissible change (assuming the software allows such changes) of dynamic operations on a file.

By supporting these policies, we are enhancing service capabilities, which in turn satisfy the business requirements. Traceability data need to be dynamically generated and captured while the service is operational. The delay between the service invocation and traceability generation should be minimal, and is made available directly to the policy engine, to be used for validating the policies.

It is important that the service is able to describe its traceability with sufficient level of domain specificity and modelling of constraints using policies that are not too complex. At the same time its execution will not introduce too many overheads, and should provide a good level of performance which can easily be integrated with the service.

2.4 Summary

In this chapter, we have provided an overview of the commercial cloud service 'ConfidenShare', and presented a number of service requirements for the evolution of the service. From these we have extracted and presented them as abstract policies.

The concrete modelling and execution of these policies are highly dependent on the generation of traceability data and integration with the service. It is therefore imperative to have a traceability model that is sufficient domain specific to conceptualize the execution of the service, as well as a policy language that is able to leverage on traceability data for its validation.

Chapter 3

Background

This chapter provides a detailed analysis of the state of the art literature on data traceability (provenance-based) for the cloud, traceable sharing of data, policy languages and frameworks for provenance.

3.1 Traceability

The earliest known attempt to provide a definition for traceability in the field of software development dates back to 1978 [84] "traceability is a property of a system description technique that allows changes in one of the three system descriptions- requirements, specifications, implementation - to be traced to the corresponding portions of the other descriptions. The correspondence should be maintained throughout the lifetime of the system".

In the food industry, traceability has a more concrete meaning, defined by EU law 178/2002 [1] as: "the ability to track any food, feed, food-producing animal or substance that will be used for consumption, through all stages of production, processing and distribution". The importance of traceability is paramount in mitigating any potential risks to food, ensuring that it is safe for human consumption [2].

Traceability also has gained interest in the design process in engineering and other such industries. Within aerospace engineering [94], it is defined as "traceability is a mechanism by which that creative process can be captured for future reference".

On a different note, retaining traceability for a design process in a military based system; whereby ability to 'trace' the design features are vitally important. Given the scale of investment and complexity of military products, failure to meet the operational expectations could be very costly [94]. However, the paper states that the capturing of design traceability can be relatively challenging. Much of traceability data generated during the early analysis phase can get lost and may not factor in the design decisions.

From the industrial perspective, the need for enhanced trust, based on data traceability in the environment such as the cloud, is a key to building customer's confidence, especially with the prospect of providing accountability (taking ownership and responsibility for issues and problems) and monitoring. From a technical perspective, it also allows users to understand how and where things have been derived from, how the transformation of data took place, and most importantly, we can trace from origin to completion [203].

Data traceability as digital evidence is of a high importance for the future of computer law enforcement. Knobler [106] presents the existing cumbersome manual process of data gathering via the method of data recoverability. However, as more and more devices are becoming interconnected, the proliferation of online computer crimes, such as child pornography and cyber-bullying, are on the increase. It would require much more robust evidence gathering techniques that are scalable, concrete and indisputable in tackling such issues.

More importantly, there is an explicit need for data traceability in cloud forensics. Cloud forensics define five areas of usage [188]: investigation, troubleshooting, log monitoring, data and system recovery, and due diligence/regulatory compliance. Traceability data can play major roles in many of these areas. For example, during the investigation, traceability data can be used to identify policy violations and investigate suspect transactions and operations. In the troubleshooting area, it can be used to determine root cause of an event, trend spanning multiple events, trace and assess current events. Log monitoring & regulatory compliance can be used for auditing and regulatory checks.

3.1.1 Traceable Sharing of Data

Models for data sharing have been studied for a number of years [179], [177], [76], [87]. Whilst the studies concentrate on the sharing of data over the Internet, they do not focus on the running context or provide the means of capturing traceability of sharing the data.

On the mobile platform, 'Traintdroid' is an initiative proposed by Enck *et al.* [70]. Their approach is based on using tags to track sensitive data used by various applications and provides a real-time analysis by leveraging on the runtime of Android execution environment. It provides various levels of tracking; starting at the highest level, then going to message level tracking, variable level tracking, method level tracking and finally to file level tracking. This allows a greater level of granularity (depending on the sensitive nature of data, from a finer level of granularity at the top to coarse grained at the bottom).

The tracking is done by first defining sets of taint tags in the virtual taint maps by the trusted application. The taint tags are then propagated with the data flow rules. Any

new instances of the interpreter propagate the taint tags, and calls from the untrusted app on tainted tags will raise an alert.

The advantage of this model is that it can track real-time service invocation based on the tainted tags, with a modest level of overhead of 14%.

The drawbacks are:

- It is specific to an Android platform, which requires modification to the kernel; this may cause interoperability issues with other Kernels.
- The tracking is non-persistence; it is unable to retain historical information of tracking.
- The tracking is only for data flow; it does not consider service flow.

Sharing data in a constrained environment (meeting) has been investigated in a multi-disciplinary project called Mematic [39]. It uses an open source tool called Access Grid (AG) [51] for collaboration and resource management for video conferencing. This tool works collaboratively with another main driver called Compendium, which is a hypermedia tool [135]. It is designed to capture issues, ideas and arguments in a discussion, and link to any documents or external sources. It is based on a technique called "Dialogue mapping" for creating the dialogue maps. However, it lacks capturing of provenance (data traceability).

3.1.2 Provenance-based Traceability

The term 'Provenance' comes from the French word 'Provenir', which is ultimately from the Latin 'Provenire'. The literal meaning of the term provenance, according to Oxford dictionary is: "(i) the fact of coming from some particular source or quarter; origin, derivation; (ii) the history or pedigree of a work of art, manuscript, rare book, etc.; concretely, a record of the ultimate derivation and passage of an item through its various owners." [138]. It is a well understood area in art and digital-libraries, where lineage, pedigree and source plays a major role in understanding how/where things have derived from, and in determining its authenticity and value [113]. There has been a great deal of interest in provenance with the computer science community over the last half a decade [219], [47] and has seen a steady growth in understanding and appreciating the much needed provenance for computing [98].

Data provenance is defined as: "information about entities, activities, and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability or trustworthiness" [85].

The provenance information can play a pivotal role in determining if something should be trusted or not [79], [45]. This is vitally important, especially when it comes to the Web, where masses of data have been produced and published, and sometimes can come across as contradictory or questionable [139]. Provenance is sometimes referred to as why, how and where provenance [40], [50], [144] are fundamental in answering questions such as: What processes were involved in transforming the data? Who had access to the data? Where did the changes to the data occur? Why is the expected data different from the actual data? How was the process carried out? Who is responsible for initiating the process? When did the process start and finish? Did the processes follow the necessary flow of executions? In order to answer such questions, one needs to look at how provenance can be captured, modelled and queried.

Much of the earlier work has been done for provenance is in the area of scientific workflow [24], [12], [34], [35], [58]. One of the most desirable features, is to be able to automate the generation of the provenance data from a given application/service or a workflow. An attempt by Barga *et al.* [25] to model such systems have been proposed. Their solution is based on using WinFX (Windows Workflow Foundation) [172] to auto capture provenance data using various levels with different granularities. Although the model seems to be good at capturing provenance data at multiple levels, the quality of data remains a concern. Their proposed model is relatively complex and assumes all the provenance data that are captured is complete, consistent and unambiguous. Also, it is limited to workflow models and the implementation is based on .net technologies [208].

Dietz *et al.* [64] proposes a provenance-based security mechanism for smart-phone operating system, called 'Quire'. It is designed to track IPC (inter-process communication) [80] between services and provides a signature mechanism to be used by services to sign statements that can be verified by other services on the same device. The advantage of the Quire is that it is lightweight and does not cause significant performance issues. However, it is specific to the Android platform and will not be able to operate on a locked-down OS such as iOS.

A study on provenance and compliance is presented by Ibbotson [97]. The report highlights the role of provenance for control activities, such as approval authorisations, verifications and reconciliation. The provenance-based control can improve the overall quality of the auditing process, improve transparency, however, it may lead to additional overheads to the existing implemented processes.

Moreau and Ibbotson proposes [140] standardization of provenance systems in service oriented architecture. They have identified three key areas of potential standardization options. The first area is to define standard APIs that can be implemented as libraries for providing services with provenance functionalities. The second area is to define a standardized way of recording and querying of provenance data, and finally a standard provenance data model. Such levels of standardization would enable development of

provenance-based web services that are interoperable with other web service applications.

A structure to capture provenance of a process document and to present in a graphical representation as a directed acyclic graph is presented by Moreau *et al.* [143]. It is based on an open approach that is agnostic to any technologies or processes that is used for electronic data. Such approach allows individual vendors and companies to provide their own implementations, which should be interoperable.

The Open Provenance Model (OPM) [146] came into existence as a result of the second provenance challenge [145]. OPM provides a structured way of interchanging provenance data between platforms. This model allows building of provenance data for 'anything', irrespective of being digital or non-digital. More importantly, OPM defines a set of rules that identify valid inferences that can be made on a provenance graph. It is technology-agnostic; a developer can build a custom made tool to operate on the model.

The provenance model (OPM) provides a graphical representation of provenance history as a acyclic graph, consisting of nodes (vertices) and relationships (edges). A node in the OPM can be either an agent, an artifact or a process [146].

There has been a great deal of activity around defining provenance based languages. Languages such as Proof Markup Language (PML) [176], Provenir ontology [191], Provenance Vocabulary, Dublin Core [23], WOT schema [90], SWAN provenance ontology [53], Semantic Web publishing Vocabulary [44] and Changeset vocabulary [210] have been proposed. Interestingly, each of these languages, to a certain extent, can be mapped to the core of the OPM [190]. OPM can be seen as the common subset of all these languages.

OPM, however, has been superseded by the W3C standardized model called Prov [147], [114]. The model has two distinct advantages over the previous one, namely:

- A greater number of relationships are defined to describe interactions between entities, activities and agents.
- It has support for extensibility, where domain specific properties can be added as required.

At a deeper level, Prov appears to have inherited many features from the OPM model. The number of nodes and their symbolic representation remains the same, with a subtle difference; the 'Artifact' is renamed as an 'Entity', 'Process' is relabelled as an 'Activity', and the 'Agent' remains the same. It provides a graphical symbol for each node:

An Agent - Represented as a pentagon symbol (see Figure 3.1), which is primarily responsible for triggering or initiating operations within a service, as well as responsible for the existence of an entity and other agents.

FIGURE 3.1: Prov - Agent

From a software perspective, it can be seen as an end user interfacing with a service to execute some form of functionality or to carry out some operations. It can also be responsible for the presence of a piece of data.

An Entity - Represented as an eclipse symbol (see Figure 3.2). It can be a digital, physical, conceptual or other type of a thing.

FIGURE 3.2: Prov - Artifact

In respect to software, it is a piece of data that is user/machine generated or other forms of static data (customer record, personal information, and others).

An Activity - Represented as a rectangle symbol (see Figure 3.3). It is a running process or something that carries out an operation within a service.

FIGURE 3.3: Prov - Process

An activity generally uses entities as inputs and typically produces some sort of output. In some cases, one activity may trigger or invoke other activities. The nature of interactions amongst the nodes (entity, activity and agent) are defined by edges. Prov enhances the number of edges significantly over its predecessor OPM. There are inclusions of new edges on top of the existing ones. Below are edges from Prov which are of interest to us for modelling of traceability for our service (see Section 2.1).

wasInformedBy - The wasInformedBy relation defines a generic communication that has taken place between two activities [114].
wasDerivedFrom - A derivation is defined as "transformation of an entity into another, a construction of an entity into another, or an update of an entity, resulting in a new one".
wasAssociatedWith - An association is defined as: "An activity association is an assignment of responsibility to an agent for an activity, indicating that the agent had a

role in the activity. It further allows for a plan to be specified, which is the plan intended by the agent to achieve some goals in the context of this activity".
wasGeneratedBy - A Generation is defined as " the completion of production of a new entity by an activity. This entity did not exist before generation and becomes available for usage after this generation".
wasAttributedTo - An Attribution is defined as: "ascribing of an entity to an agent. When an entity is attributed to agent, the entity was generated by some unspecified activity that in turn was associated to agent. Thus, this relation is useful when the activity is not known, or irrelevant".
used - An Used is defined as: "Usage is the beginning of utilizing an entity by an activity. Before usage, the activity had not begun to utilize this entity and could not have been affected by the entity".

An interesting aspect of Prov is the feature of extensibility. It recognizes the fact that the service modelling of each domain may differ significantly. Rather than trying to provide a model for everything, which would be problematic, the focus is to provide a generic building blocks that are common across all domains. For domain specificity, this can be achieved through extensibility.

The drawback of the model is that it takes a simplistic view of the software world. Software by nature is complex, large and highly structured. It does not define the implementation guidelines on how to dynamically extract and capture provenance data with the necessary relationships (especially a complex relationship such as *wasDerivedFrom*). The combination of nodes and edges can generate a huge structure, which can be unmanageable and scalability issues.

3.1.3 Provenance Framework

A number of provenance-based frameworks have been proposed [209], [8]. Tsai *et al.* [209] proposes a SOA (Service Oriented Architecture) [71] data-provenance framework. This framework is based on the non-standard provenance model, and entails functionalities such as: multiple data provenance classification (minimal provenance, time-based, event-based, etc.), data collection (actor-based and time-based), dynamic analysis (security policy checking service (SPEC), integration estimation service) and others. The checking source SPEC appears to have some degree of correlation with our work, however, no information is supplied in relation to the language used, supported features, limitations, and how it operates on the provenance data.

Aldeco-Perez *et al.* [8] proposes a provenance-based compliance framework, based on the Open Provenance Model. The framework provides a processing view (represented as a provenance graph for a specific execution time) and usage policy definition (UPD). It uses the UPD to validate against the processing view for compliance. The framework,

however, lacks the integration with the commercial applications, and policy standard such as XACML [119].

Kepler provenance framework [151] is a workflow management system for collecting, and processing of provenance data. It provides three APIs: recording, query and management for handling such task, as well as algorithms for tracking and finding files. While their solution works well for workflows, it is not generic enough to be used for general purpose services. Karma is also a provenance framework [198] similar to Kepler for workflow system, but does not have the additional processing algorithms and neither incorporates any support for provenance-based policy control.

3.1.4 Provenance-based Methodology

PriMe (Provenance Incorporating Methodology) is a design methodology [159], [158] used in software development design phase to make an application provenance enabled. This enables applications to integrate with the provenance architecture [86]. PriMe in essence is broken down into three distinct phases: provenance question capture and analysis, actor-based decomposition, and adapting the application.

Phase 1: Provenance Question Capture and Analysis - The purpose of this phase is to identify questions related to the provenance data for the application. Five pieces of information are required to complete this phase: the provenance question, provenance query, the formalization of the provenance question and retrieval of the answer, scope/data item and start item (a data item that is subject of the provenance query) [7].

Phase 2: Actor based decomposition - In this phase, the design is mapped to a structure based on actors which record process documentation and interactions. PriMe adopts an iterative approach to achieving the correct level of decomposition or granularity in order for the actors to correctly answer provenance questions.

Phase 3: Adapting the application - In this phase, it involves making implicit information identified in the previous phase explicit, in order for the application to record the process documentation. This will allow actors to perform queries in order to answer provenance questions.

The positive aspect of this methodology is that it is iterative based. If certain details have been missed out during the first iteration, further iterations would be able to capture the loss of information. On a negative note, three levels of iterations are relatively time consuming and can be quite complex to integrate into the agile-based development such as the Scrum [196], XP [31] and others.

3.2 Cloud Computing

Cloud is built on many existing tools and technologies, reducing the cost of service delivery whilst increasing the speed and agility of service deployment [215], [18]. The core technology behind cloud computing is virtualization [103], [211]; it empowers the whole cloud computing paradigm. The virtualization technology allows the separation of physical hardware and the operating system by creating an abstract layer between both. This allows a greater degree of flexibility by being able to share the same physical resources virtually, by more than one OS.

There are currently three well defined service models: Infrastructure as a Service (IaaS [187]), Platform as a Service (PaaS) [54] and Software as a Service (SaaS) [104].

There are several open source and proprietary cloud platforms these models run on. The leading open source platforms are: Eucalyptus [165], Abicloud [174], Nimbus and OpenNebula [197], Cloud Stack [55], [110], Open Stack [28]. They are all Linux-based platforms. Each can be deployed either as a private or public cloud [174]. The commercial offerings are: Amazon EC2 [216], Google AppEngine, Microsoft Azure and Manjrasoft Aneka [214]. One point to note that, while EC2, Azure and Aneka are IaaS, Google AppEngine is referred to as a PaaS. All these platforms are designed to scale, however from a SLA perspective, only Amazon provides 99.95% availability [3] which is on the server side, and not end-to-end. At present we are not aware of any services that provide facilities for the user to track data traceability. Other variations of clouds and frameworks have been proposed [10], [9], [118], which are still at a relatively primitive stage.

One of the earliest attempts to define an ontology for cloud has been proposed by Youseff *et al.* [220]. They define five layers of conceptual hierarchical structure. Starting with Software as a Service (SaaS) as the top layer, which describes the service level operations. This is followed by Platform as a Service (PaaS) consisting of: Infrastructure as a Service (IaaS), Data-storage as a Service (DaaS) and Communication as a Service (CaaS). The underlying layers are a software kernel, that sits top of Hardware as a Service (HaaS). Each layer details pertinent tools and technologies for empowering them. However, the ontology described is relatively abstract.

Han *et al.* [89], on the other hand, proposes a more concrete cloud ontology based on semantic Web technology (RDF/OWL) [30], [129] that focuses on three types of services: IaaS, PaaS and SaaS.

Firstly, the IaaS is composed of hardware, software and resource provisioning. Many of the sub-nodes can be extended further to encapsulate additional dependencies. For example, a file system can be described as a distributed file system, virtual file system, physical file system, etc. Amongst the important features that are substantive for companies and organizations are Service Level Agreements (SLA) [126] and Operation Level

Agreements (OLA). SLA and OLA determines the quality of service (QoS) [102] required by the companies with a level of accountability. The SLA/OLA should be incorporated into the IaaS decomposition.

Secondly, the PaaS entails the environment, testing, deployment and build features of the services. The decomposition of some of the sub-classes at the platform level does not adhere to the relational principle 'is-a'. For example, IDE (Interactive Development Environment) is-a BackupAndRecovery, or IDE is-a DataSecurity, which does not seem to logically satisfy this constraint.

Finally, the SaaS provides service related details. A large number of services have been detailed in this layer. However, not all elements are directly colligated to SaaS. For example, the element 'Network' consists of Proxy and DNS [32]. They are more closely associated with the running context of the services, hence more applicable in the IaaS layer.

The ontology proposed by Youseff *et al.* [220] is much more prudent and more complete than the previous one by Han *et al.* [89]. The distinctive roles of service models may not necessarily conform to the accepted cloud standard definitions [132] and does not really model the sharing of cloud resources.

An interesting concept of ontology-based resource management for cloud has been proposed by Ma *et al.* [122]. They define a generic cloud based ontology with six distinctive sub-classes: CloudComputingResources, CloudService, SLA, CloudJob, CloudUser and VirtualMachineManager. The ontology is used to allocate algorithms based on the user requirement. The advantage of this approach is that it allows dynamic allocation of resources based on user needs. However, it lacks in the following:

- A user can request for a job to be executed based on certain requirements (resource, budget, OS, etc.) but it does not provide the necessary means to validate or verify how the execution occurred.
- Once a job request has been defined, it does not facilitate declaring of policies or rules to cater for additional controls.

Deng *et al.* [63] presents a cloud-based ontology which provides a formal modelling of service offerings and its associated processes. This model demonstrates knowledge of 'composibility' of services. It represents the relation between the service offering, constraint/rules and ways to implement the offering, this is very much analogous to the OWL-S [41] for Web services. The composability offering is either a simple or composite offering. The simple construct is based on a data system (DB2 [62]), whereas the composite is Java/J2EE [6] offering. The ontology is currently restricted to these technologies, however it should be adaptable enough to incorporate other technologies as well.

The Glue ontology is an initiative proposed by the OpenGridForum [16]. The ontology provides a comprehensive model of the Grid based system [77]. It explicitly defines, among other host characteristics, Grid entity and protocol, which are specific to grid-based services. The ontology at the abstract level may be ported to cloud to describe its infrastructure, platforms and services. However, not everything in the ontology would be relevant or can be matched. Conversion is generally prone to errors and misfits, therefore, it would probably be simpler to create a cloud specific ontology.

3.2.1 Provenance for the Cloud

Today, *one important piece of the jigsaw missing from commercial cloud is provenance.* Various models/algorithms [222], [52], [88], [116] have been proposed over the last few years. Their work ranging from use of rules to track data provenance (from the creation of a data to its current state), dynamically securing cloud storage with provenance (that uses provenance as part of the group signature to aid the reconstruction of tags to identify of misbehaving user), provenance feedback in cloud services, to data forensics based on provenance. Moreover, a number of studies have been carried out [99], [81], [4], [192] during the same period. Studies include: application scenarios of provenance data, challenges of provenance and its management in the cloud and provenance benchmarking (which can aid in identifying bottlenecks in the performance, scalability and robustness of the system). Much of what is stated are hypothetical scenarios and use cases, however, practical applications of them may vary.

Macko *et al.* [123] proposed an approach for collecting provenance via the Xen Hypervisor [26]. Xen is an open-source level-one hypervisor similar to the commercial offering of VMware ESX [154] virtualization solution. It currently empowers the leading EC2 cloud solution by Amazon. Their approach is an extension to the existing PASS [156] solution which has the following deficiencies: Revision of the kernel requires the reintegration of PASS changes, and use of stackable file system makes it difficult to collect provenance on root volumes. Their proposal is to have a separate VM running the host OS (DomU, para-virtulized) and intercept the DomU's system calls by placing a hook in XEN's syscall_enter mechanism. This enables the system to collect provenance related information such as: sys call no, ref string and struct, current working directory, taskGroupId, and others. This is then sent to the PASS analyzer to extract relevant provenance information. *The analyzer only seems to remove any duplications and ensures no cyclic dependencies occurs.* One advantage of this proposed architecture is that the intense processing requirement of the analyzer can run separately on a different VM, which can enhance overall performance of the application and in theory should scale.

The disadvantages to this approach is that firstly, the interceptor requires a large ring buffer to store the provenance information before it is consumed by the analyzer. This will consume large amounts of memory and buffer operation which typically reduces the

performance of the CPU. Secondly, the interceptor can only see the file paths as strings, a path may have multiple links (symbolic, physical) thus multiple paths can refer to the same object.

Apart from technical challenges, we also see a further obstacle to capturing provenance in this manner. The interceptor can end up picking up a lot of system calls, which may not be relevant to the user's application needs. It does not differentiate between good and not so good provenance. As a consequence one may end up with a gigantic unmanageable of provenance data.

Muniswamy-Reddy *et al.* [157] on the other hand try to address similar issues of automation of provenance collection, by proposing three protocols for storing provenance for their existing cloud service. The provenance data are collected using their existing system called PASS (Provenance Aware Storage System) [156]. Any objects stored in the system automatically extract the provenance data related to it, for example a system call read, write, etc.

With the first protocol, both the provenance and actual data is stored in the same cloud object store (S3) [169]. The second protocol separates the data object and associated provenance by using a separate database called SimpleDB [68], that is used for the provenance storage. This solution overcomes the restriction of the metadata object size, and can provide efficient queries, because it can retrieve indexed provenance from the SimpleDB. The final protocol is built on the previous protocol by adding a messaging service. The cloud messaging service (SQS) and transaction ensures the provenance data coupling.

These protocols ensure the provenance data coupling (consistency between data being recorded and its provenance), multi-object causal coupling (maintain cause and effect relationship), data-independent persistence (provenance data is persistence, even after the object no longer exists) and efficient query are achieved. The drawback is that they are specific to the PASS solution, and it may be difficult to reuse or integrate with other provenance models, such as the OPM [142], Prov [136], PML [59], etc.

Using of provenance at the infrastructure level for repeatability and curation of scientific computing results has been proposed by Lampoudi [112]. The focus is on using VMIs (Virtual Machine Image) to recreate the computational environment for reproducing the results. While the research looks promising, further work is required in standardizing the metadata model (cloud provenance model) for such need.

3.3 Policy Language

In the domain of software engineering, policy languages are primarily designed to handle access control, which plays a pivotal role in safeguarding systems from unauthorized access and providing different levels of access granularity. There have been several language specifications developed over the years. Notably, one of the more well known and earliest was ACL (Access Control List) [194], used primarily in the UNIX OS. It centres around creating simple relations between users and resources in order to define rules. However, this can result in a large collection of rules, that can be cumbersome to manage (especially if the user status changes).

This drawback led to the development of the Role Based Access Control (RBAC) [74]. RBAC assigns permission to roles, then roles are assigned to users [73]. There are two distinct advantages of such an approach. Firstly, through role inheritance, it avoids duplicate definitions. Secondly, separate duties prevent users being assigned mutually exclusive roles. On the downside, occasionally differentiating the roles in different context proves to be difficult; in some cases more roles than users were produced. RBAC is typically a coarse grained approach, where the primary focus was permissions only, although later derivations were developed to handle access control as well [27], [152].

Attribute based access control (ABAC) [221] was born out of deficiencies of the RBAC [195]. The core principles are based on using user attributes to grant access. This approach is seen as very flexible in terms of expressing rules, which can be very fine grained. For example, the role itself can be considered as an attribute. However, on the negative side ABAC operates at the application level. Each application has its own set of policies, which can be difficult to administer, and may lead to conflict in policies [204].

The policy based access control (PBAC) was next to be introduced [223], [182]. It takes into account attributes, but at the enterprise level (not application level). This is particularly useful for regulatory compliance purposes, but introduces another issue. A possible conflict of centralized rule attributes may not correspond to local rules. This may not be a big issue in a non-dynamic environment, where things remain fairly constant over time, and both enterprise and local can be synced to ensure such issue does not occur. However, languages such as Ponder [121] and KAoS [141] have been proposed to be adaptive to the dynamic environment.

XACML (Extensible Access Control Mark-up Language) [82] is a general purpose policy language (combination of ABAC and PBAC) that defines XML based syntax and architecture for deploying them. Figure 3.4 shows the XACML 3.0 architecture [185], which is based on a modular structure. It defines the data flow interactions between the modules. The interactions are as follows:

Starting at the PAP (Policy Administration Point) module, this is where the policies

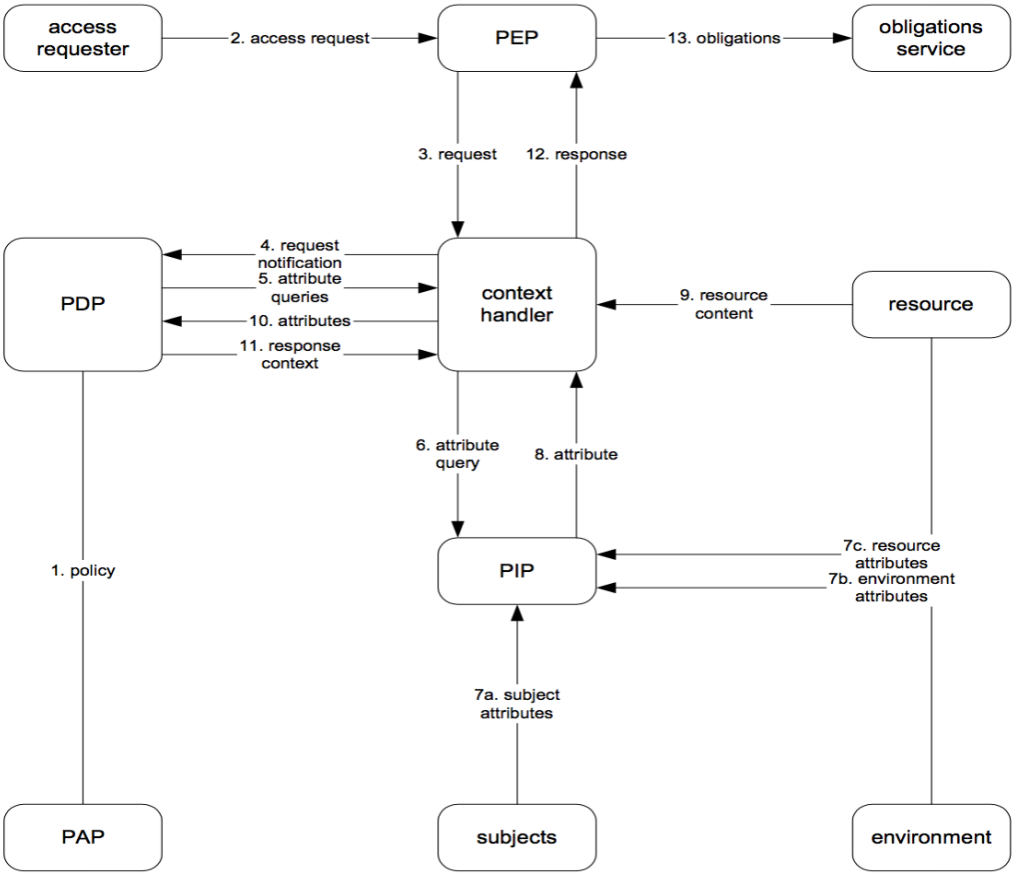


FIGURE 3.4: XACML 3.0 Architecture

are written typically by an administrator. The service level interaction starts at "Access requester", where a request is sent to the PEP (Policy Enforcement Point). This is then sent to the "Context Handler" in its native request format (XML). The "Context Handler" interfaces with the PDP (Policy Decision Point) which is responsible for evaluating the request for making an appropriate decision. It provides all the necessary information required by PDP to make the correct decision. Once the decision is formulated and received, it is sent to the PEP module. The PEP module evaluates the response, and interacts with the "Obligation Services" to fulfil the obligations [185].

The advantage of XACML is it removes the complexity from the application/service and allows administrators to handle the policy needs and requirements, as well as being both ABAC and PBAC. It is a widely used standard in the industry, and is considered to be a relatively mature standard. The drawback is that the decisions are purely made based on the data; it does not take into account any historical metadata (provenance data) associated with the data.

There are other Web specific policy languages such as WSPL [14], WS-policy [22] and KAOS [213], designed to express policies related to the security, transition, reliability of communication and others. WS-policy is primarily focused on allowing end-points to specify requirements and capabilities for establishing connections, however, it is not so

suitable for complex application specific policies [109]. WSPL on the other hand can handle complex application specific policies. The language is inspired by XACML and inherit features such as name-value paired and primitive data types for attributes [14]. KAOS is designed to support the policy requirements for Semantic Web Services.

On a different note, Deontic Logic [133] is a field of logic that handles obligations and permissions that can be integrated into a policy language. However, there may be issues related to how to properly represent conditional obligation [67] in Deontic Logic.

Amongst these policy mechanisms, there are two distinct features which are in common; firstly, they all rely on data (non-provenance) to evaluate their policies. Secondly, the policy declarations are static. In other words, all the policies are created first, and then are acted upon. However, the volatility of distributed environments such as the cloud, where environments are constantly changing, requires policies that can adapt to the contextual changes dynamically and take provenance data into account.

3.3.1 Provenance-based Policy Languages

Provenance based policy language is an emerging area of access control that uses the ancestral information to determine a course of action to be performed, which can be in the form of various level of access rights. There has been a number of provenance based policy languages proposed [170], [15], [206] with varied levels of constraints and limitations. These languages either proposes or uses provenance data as their source for their policy validation.

Park *et al.* [170] presents provenance based access control that uses OPM model and define policies based on their own policy language. The language is able to express policies based on the nodes and edges of provenance elements but not their associated properties, and it does not map to industrial policy standard (XACML).

Anderson *et al.* [15] proposes a potential provenance-aware configuration language and access control mechanism. This language is seen as important for securing and audit changes to configurations.

Doganata *et al.* [65] proposes a model for authoring and deploying business policies dynamically for compliance monitoring. The provenance model is proprietary and specifically designed for business related applications. The language does not capture the provenance of policy decision making, so there is a loss of provenance information. It does not provide any mapping or integration to XACML standard.

Stepien *et al.* [204] on the other hand, proposes a human readable form of a policy language that is based on a well known standard XACML [5], [13]. A policy can be defined easily using natural language, which is then converted to XACML format. However, it does not cater for provenance data.

PAPEL [184] is a provenance-aware policy execution language. The language tries to integrate the popular XACML general purpose policy language with the provenance model called OPM, albeit a relatively loose integration since it only uses a "Step" primitive to represent a single processing step which depicts a process in OPM.

Step (Data, Actor, InvolvedAgents, Category, Purpose, ID, PIDs) ...

Step only defines primitive parameters. This can restrict the expressibility and extensibility required for modelling complex provenance structures (Prov). Beyond capturing a step, it does not have a natural way of expressing relationships that exist between processes, entities and agents. However, it may be possible to encode such information using attributes, which can be tedious and cumbersome.

Syalim *et al.* [207], proposes an access control method for provenance, based on a direct cyclic graph. Their approach is to define policies within a relational database that operates on nodes, edges and paths of a graph. The nodes, edges and paths are modelled as three tables in the database. A separate table is used to define a list of access controls for various users for provenance information (nodes, edges and paths). The access control is coarse-grained (supports grouping), but it lacks the flexibility to define policies based on attribute access control (cannot define policies operating at the property level of nodes and edges).

Cheney [49] gives a formal model for security control for provenance, and Martin *et al.* [127] provides pertinent details of the applicability of provenance as a security control.

A graph grammar approach for rewriting redaction policies over provenance graph has been proposed by Cadenhead *et al.* [43]. The redaction approach allows hiding of sensitive information in provenance graph by rewriting it with relevant information. Their work has been extended by Danger *et al.* [61] by proposing ACPL language, which introduces a transform construct to support new query evaluation strategy.

Ramane *et al.* [181] proposes a provenance-policy based access control model for data usage validation in the cloud. The provenance model is based on a relational schema using four tables. The model is designed to operate in the cloud, but provides limited information as to how the provenance data are generated, and handled amongst multiple virtual machines. The policy generated provides limited access control using users data, and since the provenance model is based on a relational model, it is limited in its expressibility of provenance for the cloud.

Nguyen *et al.* [163] also proposes a provenance-based access control for the cloud at the IaaS (Infrastructure as a Service) level. They have chosen OpenStack as the main IaaS platform. It defines its own non-standard policy language [60] to operate over provenance data generated (their publication does not make it clear the nature of the

provenance data captured) at the infrastructure level, based on the OPM model. Their earlier work involves extending of XACML architecture for enabling provenance-based access control [162]. The extension is based on creating a new XACML function which executes provenance related query (SPARQL). This is a good approach to build on.

Bates *et al.* [29] on the other hand presents a provenance-based access control for cloud environment, which focuses more on the Software as a Service (SaaS) model. It uses its own non-standard provenance modelling as well the policy language, which may not be directly compatible with the standardized Prov model, and industrial standard XACML.

Ni *et al.* [164] proposes an access control language for a custom provenance model. The language is inspired by XACML and composed of four key sections: Target, Condition, Effect and Obligations. The target defines the subject and record for the policy to apply, as well as the restriction and scope. The condition on the other hand represents a boolean expression that presents additional contextual related requirements. The effect determines the course of action to be taken (absolute permit, deny, necessary permit and finalizing permit). Finally, obligation defines operations to be performed before conditions are executed. The language is well integrated with the database-based provenance model implementation. The database defines three tables for provenance (node, edge and path) and a single table for access permissions to the users. However, the model itself is not standardized, and does not provide enough extensibility to be able to apply to other domain services such as the cloud. Also, the policy language does not cater for resources with arbitrary path length. Each resource must be explicitly identified beforehand rather than being matched against a provenance graph. To overcome this issue, T. Cadenhead *et al.* [42] proposes an extension to the language with regular expression grammar, to operate on provenance graphs (OPM). This allows the policies to take OPM's node and edge names into account when declaring policies.

An implementation is provided that parses a policy into a regular-expression query, which is then executed as SPARQL to be operated on a RDF graph (OPM) is demonstrated by PrudHommeaux *et al.* [178]. This work is very much analogous to our proposed work, however, our approach improves in the following areas: Firstly, their policy declaration using XML is not fully coupled with the OPM model (does not support Prov and difficult to define properties associated with nodes and edges). Secondly, their policy language does not define rules, therefore a policy is likely to be relatively large and complex, which can affect the performance time, and likely to be prone to errors. Thirdly, the provenance of policy execution is not captured or recorded. Fourthly, the declaration of the policy values is static and does not accommodate dynamic policy values. Finally, it is not designed to run within an existing XACML policy engine.

Pignotti *et al.* [175] proposes a toolkit for developing provenance-aware application. The toolkit facilities for provenance creation, visualization, repository with various ontologies (OPM, SIOC [37], FOAF [38]) and policy reasoner. The policy reasoner is based on SPIN

[78] ontology which is a SPARQL-based rule and constraint language for the Semantic Web.

From all the provenance-based policy languages presented above, we can see summarize our findings as follows:

- All the policy languages discussed in this section are using or trying to use provenance data for validating their policies. While some use their own provenance models, majority are targeted for OPM model.
- The syntax and semantic of languages vary; some use regular expressions to express their policies, others are XML based. However, they still lack the close integration with the OPM model (majority do not support Prov); primarily defining policy conditions, that not only be able to express on the nodes and edges, but on their properties as well.
- There seems to be two emerging areas of research directions for provenance-based policy languages. Firstly, toward provenance-based policy language for the cloud. Secondly, toward commercial-based services. Both which are highly complementary to our research. We do believe in the longer run, the both strands would converge with the increase maturity of the provenance, as they are highly complementary to each other.

3.4 Summary

From the literature review, we can see traceability as being an important area which has been studied in various fields. Provenance-based data traceability is of particular importance to the service requirements (see Chapter 2), since it can be used to determine the data origin, dependencies and others. Provenance has been studied for a number of years, and currently there are different flavours of provenance model primarily designed for generic environments; of which the standardized and leading model is Prov. However, for our service requirements, Prov is fairly generic, and requires a model that is more domain specific to capture the necessary cloud context related information. A limited number of papers proposes provenance for the cloud [123], [112], [157]. However, they are primarily focused at the infrastructure level, and do not define a comprehensive cloud provenance model that is implementation agnostic and non-vendor proprietary.

On top of the traceability model, our service requirements require expressing of business rules, validation and enforcing them using traceability data. Our literature review shows a number of policy languages that exist which are primarily focused on non-provenance data, of them, the most widely used in the industry and most mature standard is the XACML. Emerging languages such as PAPEL [184] and ACL [42] make a good attempt

at integrating provenance as part of their policy control. However, they still lack a close integration with the Prov model, and leverage on the industrial standard (XACML), and without modifications are fairly limited.

We propose to address these constraints with in the next few chapter, by proposing a traceability model for cloud services that extends the Prov model. On top of that, a policy language based on the Prov notations that can use traceability data to validate its policies. As well as a framework that maps the policy language to industrial standards such as XACML and provides necessary stacks to integrate with cloud-based services.

Chapter 4

cProv - A Traceability Model for Cloud-based Services

Having traceability of a running service in the cloud environment is critical in keeping track of all the online activities that have taken place. This vital information thereby gathered can provide us with an end-to-end view of the internal workings of any of the cloud based service in their order of occurrence. Obtaining such information can have several commercial benefits too, especially for the service provider. For example, it can be used to monitor the status or health of a running service, to rapidly identify any emerging problems, to determine issues and for a more robust auditing. Advance handling of traceability data can support searching for trends and patterns of the end user's interaction with the service, which in turn can potentially be used to deliver a more personalized and targeted service experience.

However, a number of questions need to be answered first in order to achieve all of this. Firstly, how do we achieve this traceability? Secondly, what information is required to ensure the flow of the data and to ensure that the process invocations are all captured in a standardized way? Finally, how can all the captured information be utilized to add value to the service offering?

Our contribution in this chapter is as follows: *A provenance-based cloud traceability model that captures the running characteristics of a cloud service. It extends the standardized Prov [148] model by defining sub-types of the core concepts to be more precise by catering for the specificities of cloud-based services. This enables the service to describe its provenance more concretely.*

4.1 Motivation for a Traceability Model

In a cloud service [92], resources may be transmitted from various, often remote sources, such as: a PC, laptop, mobile and other devices. What is a 'resource' in the context of the Cloud? This may be explained as the data which is residing outside the cloud which is then referred to as a *'digital resource'*.

The Cloud has an extra layer of abstraction, commonly referred to as a virtual layer. Once the data has been transferred into the cloud itself, we then refer to this as the *'virtual resource'*.

In general a resource [56] can be of various digital forms including the following: Audio (MP3, WMA, AAC, etc. [111]), Video (MP4, MPEG, H.265, etc. [218]), Text (ASCII, Unicode [46], [57]), and Image formats (GIF, PNG, JPG, etc. [134]).

Resources within the cloud (i.e. virtual resources) can be shared, modified or deleted by one or more participants, within a service or amongst multiple services. Such intake of operations can require the traversal of resources between multiple localities with various jurisdictions, which may not be transparent to the user. It is therefore essential to model such flow of resources. With respect to our service, achieving traceability requires satisfying of the following requirements:

Requirement 4.1 : Modelling of Service Level Operations

The execution of service operations may take place in one or more virtual servers from single or multiple physical machines. An operation may be invoked from an external source, such as a user or a remote process, alternatively from an internal source, such as the likes of a machine agent (e.g. M2M) or other processes. Modelling of service level operations provides a better understanding of the nature of invocations and the order in which they took place.

Requirement 4.2 : Modelling of Locality of the Operations

Physical servers running operations can be located in multiple localities. Processes may be sensitive to the boundary of their physical location and may be restricted in various way such as their scope of operation and thus require to be handled differently. Modelling of locality information related to process execution would enable us to determine contextual changes of the running process.

Requirement 4.3 : Modelling of Service Level Resources

A resource may be generated, produced and processed from within and outside of the cloud, with varying characteristics. Modelling of the service resources would enable us to have a better understanding of their dependencies with their associated processes (especially the inputs and outputs).

Requirement 4.4 : Modelling of Locality of the Resources

Resources can have various levels of sensitivity based on the nature of information they contain, and may require handling differently for various locality. Modelling of the geo-location information of the resource would provide insight into the flow of the data in the cloud environment.

Requirement 4.5 : Grouping of Operations and Data

In a cloud service, satisfactory execution of business logic (event) may require invocation of multiple processes and generation/usage of resources. Modelling the grouping of such invocations would enable identifying and analyzing the outcome of the individual event that was run.

With respect to all these properties, having a common traceability model which is domain specific to capture the service level traceability in a comprehensive way is profoundly important.

4.2 The cProv Traceability Model

From the literature review, we can see 'Provenance' is a concept that enables modelling of historical information in a highly structured way, typically using graph representation. Provenance by definition enables us to see the historical interactions with a service, preserving the execution ordering all the way from the origin. Prov [148] is a generic data model for conceptually modelling provenance of a service, it is not specific enough to describe the details of a cloud service in domain-specific terms. To address this, we propose a new traceability model (cProv) based on the standardized Prov model.

We define the cProv traceability model as an extension of the existing provenance (Prov) model by defining concrete sub-types of its core elements and relationships. With these extensions, cProv can support the additional domain specific requirements (see Section 4.1), and remain compatible with the Prov standard.

Figure 4.1 shows two models: Prov (top), and cProv (bottom). Prov defines the root elements and edges, which are used as the base for the sub-type elements and edges of

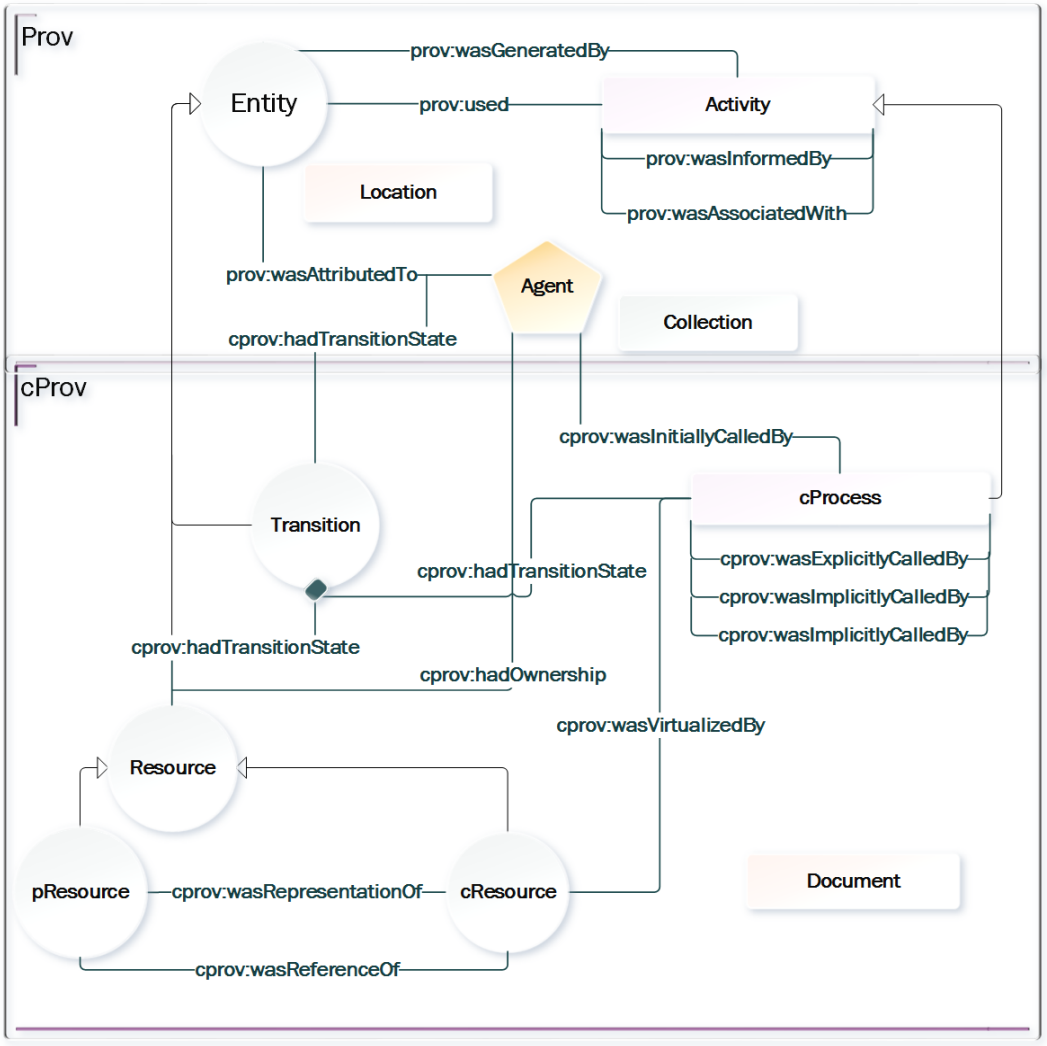


FIGURE 4.1: cProv Traceability Model

the cProv model. Note that no new provenance top level types (see Figures 3.1, 3.2 and 3.3) have been introduced to preserve compatibility.

We now concretely define all the elements and relations, and present them using Prov-N notation [147] with examples. Prov-N notation is favoured for abstractly presenting the cProv's elements and relations since they are sub-types from the Prov model.

4.2.1 cProv Elements

The cProv elements provide a greater degree of specialization for cloud-based services while retaining compatibility with Prov. From the cProv model (Figure 4.1), a total of five new elements are introduced: four of which are concrete (Transition, pResource, cResource and cProcess) and one is abstract (Resource). All the concrete elements except cProcess and Transition inherit from Resource, which in turn inherits from the top level 'Entity' element. The Transition element directly inherits from the Entity

element. Element cProcess inherits from the base level Activity.

Various namespaces have been used (see Table 4.1 below) by the cProv model to qualify names belonging to the same group of elements and their properties.

prefix	**suffix**	**description**
prov	http://www.w3.org/ns/prov#	used for the base level Prov elements and its attributes
cprov	http://labs.orange.com/uk/cprov#	used by the extended cProv elements and its attributes
cprovd	http://labs.orange.com/uk/cprovd#	used by the data values for some of the elements and attributes
ex	http://labs.orange.com/uk/ex#	used as the identifier (IDs) for each element. A service can choose to use its own

TABLE 4.1: Traceability Model Namespaces

Definition 4.1. A **'Resource'** is a generic piece of data, which can be in a variety of digital formats.

It has various properties which are common to its sub-types; it is abstract since it is not intended to be instantiated on its own.

entity(ex:id, [prov:type='cprov:Resource', cprov:resType='', cprov:trustDegree='1.0', cprov:restrictionType='', cprov:des=''])

- ex:id - unique identifier of the Resource (mandatory)
- prov:type - extensibility point of Prov (mandatory)
- cprov:resType - type of resource (e.g. data) (optional)
- cprov:trustDegree - the trust degree indicates the reliability of the resource (optional)
- cprov:restrictionType - type of restriction on the resource, a choice of cprovd:confidential, cprovd:restricted, cprovd:general and cprovd:non-modifiable (optional)
- cprov:des - brief description of the resource (optional)

The ex:id property uniquely identifies the Resource element and must contain a unique value. Prov supports extension via the 'prov:type' property (here we define the type as a 'cprov:Resource'). The cprov:resType is used to indicate the nature of the resource.

The cprov:trustDegree denotes the reliability of the resource. In the context of the ConfidenShare service, it expresses the trustworthiness of a resource. The service determines its value, which can range between 0-1; 1 being most reliable, 0 as unreliable. It is possible that over time the reliability of data may increase or decrease due to other dependencies or newly acquired knowledge. Through usage, modification or amalgamation, the trust degree value may change, and this needs to be reflected in the trust value. However, since traceability data are immutable, it is not possible to update the existing element. A new instance of the element must be created with the updated cprov:trustDegree value. Note, the trustworthiness feature of the ConfidenShare service is not part of the current development.

In the context of the cloud, a resource can have different levels of access restrictions. The attribute cprov:restrictionType is an important attribute for the service requirements, because it defines the kind of restriction that exists on a resource. Four possible restrictions are permitted: cprovd:confidential, cprovd:restricted, cprovd:general and cprovd:non-modifiable. When a resource is created, the default restriction type is cprovd:general.

Note, not all the base element (prov:entity) properties are presented here, but can be found in the Prov data model documentation [148].

The first sub-type of the Resource element is the pResource.

Definition 4.2. A **pResource** is a digital resource which resides outside of the cloud.

A pResource is described as:

entity(ex:id, [prov:type='cprov:pResource', cprov:resType='' , cprov:trustDegree = '1.0', cprov:restrictionType='' , cprov:originIP='', cprov:macAddress='', cprov:host='', cprov:des=''])

- prov:id - unique identifier of the pResource (mandatory)
- cprov:originIP - IP address of the current residence (optional)
- cprov:macAddress - MAC address of the machine (mandatory)
- cprov:host - the type of host, e.g. cprovd:PC, cprovd:server, cprovd:laptop, cprovd:mobile, etc. (optional)

A pResource is where a user acquires a piece of data from a third party (i.e. the Internet or received electronically), or was generated on the host machine (laptop, PC,

and others). In both cases the data is outside of the cloud service environment. It is important to record information such as the IP & MAC address, host and others to determine its origin. Note not all the inherited properties are listed as they have been presented in their base element.

An example of pResource element is shown below.

```
entity(ex:e002, [prov:type='cprov:pResource', cprov:resType='cprovd:text',
 cprov:trustDegree="0.7" %% xsd:float, cprov:originIP= "192.165.123.21",
 cprov:restrictionType = 'cprovd:general', cprov:macAddress="00:0E:00:02:E0:00:0A:00",
 cprov:host='cprovd:device'])
```

This example shows a resource which is of type 'cprovd:text', with an id ex:e002 and has a trust degree value of 0.7.

The second subtype of the Resource element is the cResource.

Definition 4.3. A **cResource** is a cloud resource, or more concretely a virtualized resource, which has been generated or transferred into the cloud.

entity(ex:id, [prov:type='cprov:cResource', cprov:resType='', cprov:trustDegree='1.0', cprov:userCloudRef='', cprov:vResourceRef='', cprov:pResourceRef='', cprov:restrictionType='', cprov:isReplicable='', cprov:TTL='', cprov:des=''])

- prov:id - unique identifier of the cResource (mandatory)
- cprov:userCloudRef - the reference to the user's cloud environment (mandatory)
- cprov:vResourceRef - the reference to the virtual resource in the user's cloud environment (mandatory)
- cprov:pResourceRef - mapping to physical address in the user's cloud environment (mandatory)
- cprov:isReplicable - can be replicated (for backup purpose) (optional)
- cprov:TTL - time-to-live; a period before the resource is obsolete or expires (optional)

The core properties of the cResource element are the mappings between the cprov:userCloudRef, cprov:vResourceRef and cprov:pResourceRef. As stated in the service description (see Section 2.1), the service allows use of multiple cloud repositories to store and share user's resources (files). Due to the cloud being a multi-tenanted architecture [66], multiple users share the same physical hardware with the aid of virtualization.

The cprov:userCloudRef is a reference to the user's cloud space. This is mapped to a user's virtual resource (cprov:vResourceRef), which in turn is mapped to the actual physical hardware.

The property 'cprov:isReplicable' defines if a resource is replicable and its time-to-live (cprov:TTL). After the expiration of the cprov:TTL, the resource will no longer be accessible or is permanently removed. The cprov:TTL is similar to the Prov concept of "wasInvalidatedBy" which defines an edge between an entity and activity. The main difference is the cprov:TTL is primarily focused on setting an expiry time during the creation time of the resource, while the other is defined dynamically as required via an additional edge to the resource.

An example of an instance of cResource is shown below.

```
entity(ex:e003, [prov:type='cprov:cResource', cprov:resType= 'cprovd:data',
 cprov:trustDegree="0.7" %% xsd:float,
 cprov:userCloudRef="http://orangecloud/user@bob/clusterX/imageX" %% xsd:anyURI,
 cprov:vResourceRef="http://platformX/ServiceX/resX" %%xsd:anyURI,
 cprov:pResourceRef="http://ClusterX/ServerNameX/126.23.43.45/00:12:00:12" %% xsd:anyURI,
 cprov:isReplicable= "true" %% xsd:boolean, cprov:restrictionType = 'cprovd:general',
 cprov:TTL= "2014-11-16T16:05:00" %% xsd:dateTime, cprov:des="temp data"])
```

This example shows a cloud resource which is of type 'cprovd:data' belonging to the user Bob's cloud space.

The third sub-type of Resource is Transition.

This element is designed to capture details related to the state and location of entities, activities and agents from an event.

Definition 4.4. An **Event** is a modelling of a business function that groups users, operations and resources related to the event.

An example of an event: *User Bob would like to share (fileA) as a copy (fileAA) with the user Phil.*

This event consists of a number of processes (share, copy), agents (Bob, Phil) and entities (fileA, fileAA). The start of the event is Bob (sender) invoking the process share and the end of the event is Phil (receiver) receiving the fileAA as a copy. During execution of the event, the processes, agents and entities will have states, and could be based at one or more physical locations. The state may change during the execution of the event. Such information is modelled using the Transition element (see Def 4.5 below).

It is possible for an event to have multiple execution of processes in various locations (the location details are modelled using the transition element), such as on the client side,

server side or both and can use or generate multiple resources. The event Id is included as part of the transition element. It is therefore conceivable for multiple transitions (see below Def 4.5) belonging to the same event to have the same event Id. While for server side processing, the location information would not change very frequently; we envision this to be more dynamic on the client side. A client can have multiple form factors (such as mobile, tablet, laptop and others). Each device is equipped with sensors such as GPS, WiFi, 3G/4G, which can be used to determine the locality information (geo-locations, region etc.) that is important for enforcing locality restrictions on sharing and storage or data.

Definition 4.5. A **Transition** is the state at which an operation, resource or an agent is during the course of an event.

entity(ex:id, [prov:type='cprov:Transition', cprov:state='', cprov:event='', cprov:region='', cprov:country='', cprov:latitude='', cprov:longitude=''])

- prov:id - unique identifier of the Transition (mandatory)
- cprov:region - name of the region (optional)
- cprov:country - name of the country (optional)
- cprov:latitude - parallel line geo-coordinates (optional)
- cprov:longitude - vertical meridian geo-coordinates (optional)
- cprov:state - a choice between cprovd:origin, cprovd:source, cprovd:intermediary and cprovd:destination (mandatory)
- cprov:event - id of the event this transition belongs to (mandatory)

The cprov:state attribute defines various states involved in an event execution for the ConfidenShare service. We can summarize each states as follows:

cprovd:origin - is a state where a process, data or an agent involved in an event resides outside of the cloud. This state marks the initiation of an event.

cprovd:source - is a state where the data or an agent involved in an event is residing inside the cloud. Note, if the cprovd:origin state does not exist then the cprovd:source becomes the initiation of an event.

cprovd:intermediary - is a state where one or more supplementary operations are performed, such as in a validation check.

cprovd:destination - is a state that marks the end of an event.

A transition is not instantiated on its own, but in conjunction with a process, resource or an agent, linked by a relation (see Section 4.12).

The following is an example of a transition element.

```
1 agent (ex:Bob, [prov:label="Bob"])

2 entity(ex:t001, [prov:type='cprov:Transition', cprov:region='provd:GB',
3 cprov:country='provd:England', cprov:latitude="51.5171 N", cprov:longitude="0.1072W",
4 cprov:state='cprovd:origin', cprov:event='ex:ev001'])
```

This example shows the transition information related to the agent (ex:Bob), which belongs to the event (ex:ev001) provided by the cprov:event property. Note, the relationship between an agent and transition is defined in Section 4.2.2.

The core activity element is extended by the element 'cProcess'.

Definition 4.6. A **cProcess** is a cloud process, or more concretely a virtual process running within the cloud.

activity(ex:id, prov:startTime, prov:endTime, [prov:type='cprov:cProcess', cprov:userCloudRef='', cprov:vProcessRef='', cprov:pProcessRef='', cprov:des=''])

- prov:id - unique identifier of the cProcess (mandatory)
- cprov:userCloudRef - a reference to the user instance of the service in the cloud environment (mandatory)
- cprov:vProcessRef - a reference to virtual process in the cloud service environment (mandatory)
- cprov:pProcessRef - a reference to the actual physical machine running the process in the cloud environment (mandatory)

As it can be seen from the service description (Section 2.1), a user can invoke several processes where each process runs as a virtual process (which is executed on a physical hardware). The cprov:userCloudRef property references the user's cloud space, and the cprov:vProcessRef is a reference to a virtual process run by the user. The cprov:pProcessRef is a reference to the physical hardware.

The following is such an example of a cProcess activity element.

```
activity(ex:a001, 2012-02-16T16:05:00, 2012-02-16T16:16:0, [prov:type='cprov:cProcess',
 cprov:userCloudRef="http://orangecloud/user@bob/clusterX/imageX" %% xsd:anyURI,
 cprov:vProcessRef="http://platformX/ServiceX/ProcessX" %% xsd:anyURI,
 cprov:pProcessRef="http//ClusterX/ServerNameX/ServerIpX/ServerMacX/" %% xsd:anyURI])
```

This example shows a cProcess running within the user Bob's cloud space.

These extended elements (Section 4.2.1) can effectively model the processes, resources and their context within a cloud service. However, relationships are required to provide a greater domain specific expressibility within a cloud service.

4.2.2 cProv Relations

We now introduce a number of relations that define the interactions between the processes, agents and resources (see Figure 4.2 below) within a service.

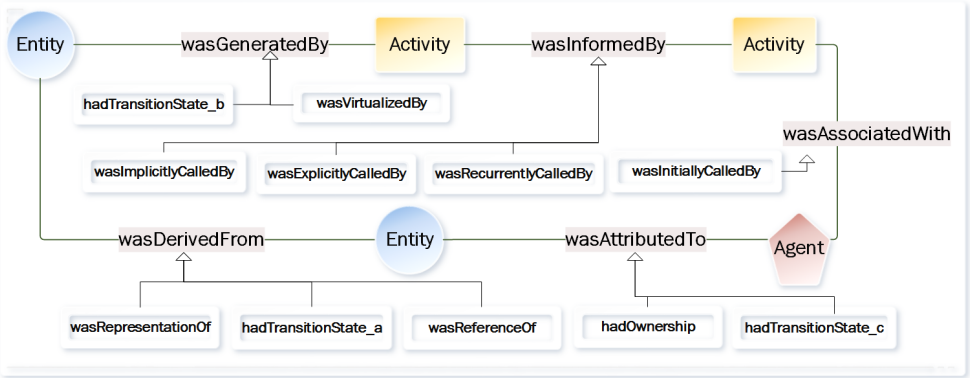


FIGURE 4.2: cProv Relationships

There are a total of nine new relations formalized to allow a greater degree of specialization in expressing traceability of cloud-based services. These relations enable us to capture the complex relationships amongst service components, and gain a better understanding of their dependencies. They inherit directly from the following Prov relations: wasInformedBy, wasDerivedFrom, wasAssociatedWith, wasGeneratedBy and wasAttributedTo.

The three relations that extend from the 'wasInformedBy' are: 'wasExplicitlyCalledBy, 'wasImplicitlyCalledBy' and 'wasRecurrentlyCalledBy'. They define the nature of interaction that takes place between cProcesses. These interactions enable us to identify if the invocation between processes are user generated (considered explicit) or manifested by the service itself (considered either implicit or recurrent). For example, a user clicks

on a button to trigger a share operation, and the sevice performs an operation such as a validation check. Alternatively, this can be a self triggered operation performed by the service (such as waiting for a response or an acknowledgement).

Definition 4.7. A **wasExplicitlyCalledBy** is a type of communication by which an invocation has taken place between two processes where a user/agent was directly involved in or initiated the interaction.

wasInformedBy(ex:id, prov:informed, prov:informant, [prov:type= 'cprov:wasExplicitlyCalledBy', cprov:explicitType='', cprov:callComm='', cprov:callMedium='', cprov:callNetwork=''])

- prov:id - unique identifier of the relation wasExplicitCallBy (optional)
- cprov:explicitType - type of call: a choice between cprovd:inform, cprovd:notification, cprovd:termination, cprovd:obfuscation, etc (optional)
- cprov:callComm - communication type, cprovd:sync or cprovd:async (optional)
- cprov:callMedium - it defines the medium of the call (cprovd:PC, cprovd:phone, etc) (optional)
- cprov:callNetwork - the network call is made from (cprovd:3G, cprovd:Wifi, etc) (optional)

The cprov:explicitType provides additional information related to the purpose of the explicit invocation. The purpose can be in the form of these operations: informed, notified, terminated or obfuscated which may be performed by the invoked process.

The cprov:callComm property captures information relating to the type of communication (sync or async) between the two processes. The cprov:callMedium and cprov:callNetwork properties capture details related to the device for the invocation and the communication network used. Capturing this information related to process invocation are important for debugging purposes. Note, the prov:informant needs to be invoked by an agent, hence an edge of 'wasAssociatedWith' with an agent should also be defined.

The following example shown below is the wasExplicitlyCalledBy.

```
1  Agent (ex:Bob, [prov:label="Bob"])
2 //see Def:4.13
3  wasAssociatedWith( ex:a001, ex:Bob, [prov:type='cprov:wasInitiallyCalledBy'])

4  wasInformedBy(ex:a002, ex:a001, [prov:type='cprov:wasExplicitlyCalledBy',
5   cprov:explicitType= 'cprovd:inform',  cprov:callComm='cprovd:synchronized',
6   cprov:callMedium='cprovd:mobile', cprov:callNetwork='cprovd:3G'])
```

This example shows the process 'ex:a001' invoking 'ex:a002' as an explicit call, with the involvement of the agent 'ex:Bob'.

Definition 4.8. A **wasImplicitlyCalledBy** is a type of communication by which a process is called by another process without direct involvement of a user/agent; usually carried out in an automated manner.

wasInformedBy(ex:id, prov:informed, prov:informant, [prov:type= 'cprov:wasImplicitlyCalledBy',cprov:implicitType='', cprov:callComm='', cprov:callMedium='', cprov:callNetwork=''])

The cprov:implicitType captures the type of the invocation that took place, for example validation, registration and others. The cprov:callCommunication defines the nature of communication that has taken place. It can be either synchronized or asynchronized.

For example, when a resource is created in the cloud, a backup copy is usually also created for redundancy purpose, and carried out implicitly by the system without the user's involvement.

An example of the wasImplicitlyCalledBy.

```
1 wasInformedBy(ex:a002, ex:a001, [prov:type='cprov:wasImplicitlyCalledBy',
2  cprov:implicitType='cprovd:validatation', cprov:callComm='cprovd:synchronized',
3  cprov:callMedium='cprovd:server', cprov:callNetwork='cprovd:Ethernet'])
```

This example shows a process (ex:a001) is making an implicit call to another process (ex:a002) for the purpose of validation (cprov:implicitType) in synchronized mode (cprov:callComm).

Definition 4.9. A **wasRecurrentlyCalledBy** is a communication by which a cProcess calls another cProcess or itself repeatedly.

wasInformedBy(ex:id, prov:informed, prov:informant, [prov:type= 'cprov:wasRecurrentlyCalledBy', cprov:recurrentType='', cprov:timeInterval='', cprov:callComm='', cprov:callMedium='', cprov:callNetwork='', cprov:repCount])

- cprov:timeInterval - time interval between each call (optional)
- cprov:repCount - number of repetitions between the both processes (an integer value, 0 indicates unlimited) (optional)

A typical case is where a process may be waiting for a notification, or is involved in an asynchronous communication.

Following example shown below is the wasRecurrentlyCalledBy.

```
wasInformedBy(ex:a002, ex:a001, [prov:type='cprov:wasRecurrentlyCalledBy',
 cprov:callType='cprovd:notification', cprov:timeInterval ="00:00:45"
   %% xsd:time, cprov:callMedium='cprovd:server', cprov:repCount="0" %% xsd:integer])
```

The wasRecurrentlyCalledBy edge between two process is primarily used to present a repetitive action whereby one process calls another process repeatedly at a certain interval. Such as, an email agent operating in a pull mode for retrieving emails from an email server.

This example shows a backup process (ex:a001) used to replicate data for redundancy purpose. It invokes process ex:a002 which provides details current changes to the service data at every 45s interval (defined using cprov:timeInterval property). This pattern is repeated indefinitely (indicated by the property cprov:repCount = 0).

The 'wasDerivedFrom' relation is extended by following relations: 'wasRepresentationOf' and 'wasReferenceOf'.

Definition 4.10. A **wasRepresentationOf** is a derivation of a resource, where the original resource is transformed by a method that retains the fundamental characteristics of the original resource to produce a derived resource.

wasDerivedFrom(ex:id, prov:generatedResource, prov:usedResource, cprov:cProcess, prov:generation, prov:usage, [prov:type='cprov:wasRepresentationOf', cprov:method=''])

- prov:id - unique identifier of the relation wasRepresentationOf (optional)
- cprov:method - the method of change of representation, cprovd:virtualization (optional)

Following is an example of the wasRepresentationOf.

```
wasDerivedFrom(-, ex:e003, ex:e002, ex:a002,-,-, [prov:type='cprov:wasRepresentationOf',
 cprov:method='cprovd:virtualization'])
```

This example shows a resource 'ex:e002' which is a pResource (resource residing outside of the cloud) is virtualized by a process (ex:a:002) that generates a virtualized resource called (ex:e003). This resource (ex:e003) is a representation of the resource ex:e002.

Definition 4.11. A **wasReferenceOf** is a derivation of a resource, where the derivation is indirectly or directly references the original resource.

wasDerivedFrom(ex:id, prov:generatedResource, prov:usedResource, cprov:cProcess, cprov:generation, cprov:usage, [prov:type='cprov:wasReferenceOf', cprov:method='', cprov:refType=''])

The cprov:method attribute specifies how the reference was accomplished, for example it can be done through an URI or a RPC (Remote Procedural Call).

An example of the wasReferenceOf is shown below.

```
1 wasDerivedFrom(-, ex:e003, ex:e002, -,-,-, [prov:type='cprov:wasReferenceOf',
2  cprov:method='cprovd:URI', cprov:refType='cprovd:direct'])
```

In this example, the resource 'ex:e002' is shared as a reference 'ex:e003' (the reference method in this case is via URI).

Definition 4.12. A **hadTransitionState** is a creation of a state that includes location and event information by an activity, entity or an agent.

We can define *three variants* of this relation for Prov encoding:

A transition is associated with an entity.

wasDerivedFrom(ex:id, cprov:Transition, cprov:Resource, cprov:cProcess, cprov:generation, cprov:usage, [prov:type='cprov:hadTransitionState_a', cprov:method=''])

A transition is associated with an activity.

wasGeneratedBy(ex:id, cprov:Transition, cprov:cProcess, [prov:type= 'cprov:hadTransitionState_b', cprov:method=''])

A transition is associated with an agent.

wasAttributedTo(ex:id, cprov:Transition, prov:agent, [prov:type='cprov:hadTransitionState_c', cprov:method=''])

The cprov:method attribute defines the transition state information allocation as statically (cprovd:static) or dynamically (cprovd:dynamic). In other words, during the start-up of the service or while running of the service.

The first example presents a cprov:hadTransitionState relation between an entity and a transition.

```
entity(ex:e003, [prov:type='cprov:cResource', cprov:resType= 'cprovd:data',
 cprov:trustDegree="0.7" %% xsd:float,
 cprov:userCloudRef="http://orangecloud/user@bob/clusterX/imageX" %% xsd:anyURI,
 cprov:vResourceRef="http://platformX/ServiceX/resX" %%xsd:anyURI,
 cprov:pResourceRef="http://ClusterX/ServerNameX/126.23.43.45/00:12:00:12" %% xsd:anyURI,
 cprov:isReplicable="true" %% xsd:boolean, cprov:TTL="2014-11-16T16:05:00" %% xsd:dateTime])

entity(ex:t003, [prov:type= 'cprov:Transition', cprov:region='cprovd:GB',
 cprov:country = 'cprovd:Wales', cprov:latitude="51.2171 N",
 cprov:longitude="0.1172W", cprov:state='cprov:source', cprov:event='ex:ev001'])

wasDerivedFrom(-, ex:t003, ex:e003 -,-,-,
  [prov:type='cprov:hadTransitionState', cprov:method=''])
```

This example shows a resource (ex:e003) and its transition details (ex:t003) linked by the relation 'hadTransitionState'.

The second example presents a cprov:hadTransitionState relation between a process and a transition.

```
activity(ex:a004, 2012-02-16T16:05:00, 2012-02-16T16:16:0, [prov:type='cprov:cProcess',
 cprov:userCloudRef="http://orangecloud/user@mufy/clusterX/imageX" %% xsd:anyURI,
 cprov:vProcessRef="http://platformX/ServiceX/ProcessX" %% xsd:anyURI,
 cprov:pProcessRef="http//ClusterX/ServerNameX/ServerIpX/ServerMacX/" %% xsd:anyURI])

entity(ex:t004, [prov:type= 'cprov:Transition', cprov:region='cprovd:EU',
    cprov:country = 'cprovd:Spain', cprov:latitude="51.2171 N",
    cprov:longitude="0.1172W", cprov:state='cprovd:source', cprov:event='ex:ev002'])

wasGeneratedBy(ex:t004, ex:a004, 2001-10-26T21:32:52,
    [prov:type='cprov:hadTransitionState', cprov:method=''])
```

This example shows a process (ex:a004) and its transition details (ex:t004) linked by the relation 'hadTransitionState'.

The third example presents a cprov:hadTransitionState relation between an agent and a transition.

```
agent (ex:ag005, [prov:label=''])

entity(ex:t005, [prov:type= 'cprov:Transition', cprov:region='cprovd:EU',
 cprov:country = 'cprovd:England', cprov:latitude="51.2171 N",
 cprov:longitude="0.1472W", cprov:state='cprovd:source', cprov:event='ex:ev002'])

wasAttributedTo( ex:t005, ex:ag005,  [prov:type='cprov:hadTransitionState',
  cprov:method=''])
```

This example shows an agent (ex:ag005) and its transition details (ex:t005) linked by the relation 'hadTransitionState'.

The relation 'wasAssociatedWith' is extended by the 'wasInitiallyCalledBy'.

Definition 4.13. A **wasInitiallyCalledBy** is an initialization of a process by an agent. This signifies the start of an event execution or a series of executions.

wasAssociatedWith(id, prov:activity, prov:agent, prov:plan, [prov:type = 'cprov:wasInitiallyCalledBy', cprov:purpose='', cprov:accessMedium='', cprov:accessNetwork='])

- prov:id - unique identifier of the relation wasInitiallyCalledBy (optional)
- cprov:purpose - purpose of the call (optional)
- cprov:accessMedium - it defines the medium of the call (cprovd:PC, cprovd:phone, etc) (optional)
- cprov:accessNetwork - the network call is made from (cprovd:3G, cprovd:Wifi, etc) (optional)

Following is an example of the wasInitiallyCalledBy.

```
 wasAssociatedWith(ex:a002, ex:ag001 [prov:type='cprov:wasInitiallyCalledBy',
  cprov:purpose='cprovd:registration', cprov:accessMedium='cprovd:mobile',
  cprov:accessNetwork='cprovd:3G'])
```

This example shows the process 'ex:a002' initially invoked by the user 'ex:ag001' for the registration purpose.

The relation 'wasGeneratedBy' is extended by 'wasVirtualizedBy'.

Definition 4.14. A **wasVirtualizedBy** is a generation of a resource that is virtualized by a cProcess.

wasGeneratedBy(ex:id, prov:entity, prov:activity, prov:time, [prov:type='cprov:wasVirtualizedBy', cprov:purpose='])

- prov:id - unique identifier of the relation wasVirtualizedBy (optional)
- cprov:purpose - the type of resource generated, i.e. obfuscated data, validated data, etc. (optional)

An example of the wasVirtualizedBy is shown below.

```
1 wasGeneratedBy(ex:e004, ex:a002, 2001-10-26T21:32:52, [prov:type='cprov:wasVirtualizedBy',
2  cprov:purpose='cprovd:validated'])
```

This example shows a virtualized resource 'ex:e004' is created by the process 'ex:a002' for validating purpose.

The relation 'wasAttributedTo' is extended by 'hadOwnership'.

Definition 4.15. A **hadOwnership** is an exhibition of belonging of a resource to an agent. The ownership can be of type: originator, contributor or possession.

wasAttributedTo(ex:id, prov:Resource, prov:agent, [prov:type='cprov:hadOwnership', cprov:ownershipType=''])

- prov:id - unique identifier of the relation hadOwnership (optional)
- cprov:ownershipType - 'cprovd:originator' if he/she is the main author, 'cprovd:contributor' he/she has contributed to it, or 'cprovd:possession' if the data is under his/her control (mandatory)

An example of the hadOwnership.

```
1 wasAttributedTo(ex:e004, ex:ag002, [prov:type='cprov:hadOwnership',
2  cprov:ownershipType='cprovd:possession'])
```

From this example we can see the resource 'ex:e004' is under the possession of the agent 'ex:ag002'.

4.3 Validating the Traceability Properties

Using the provenance model (cProv), we now demonstrate validating the five requirements (see Section 4.1) using a simple example:

User Bob creates a new resource from an existing resource that is replicated for redundancy purpose in the cloud.

For each property, a provenance graph will be used (built on the previous graph) to visually demonstrate how it has been satisfied.

4.3.1 Service Operation

The first requirement (Req 4.1) requires modelling traceability of the service operations. This is accomplished by using the cProv's node cprov:cProcess and other edges.

Figure 4.3 shows an agent 'Bob' who is an external user, invoking (denoted by the edge cprov:wasInitallyCalledBy edge) the process 'createResource', in order to create a resource. The 'createResource' process makes an explicit call to the process 'redundancy' (via the edge 'cprov:wasExplicitlyCalledBy'). This in turn makes an implicit call to the process 'copy' (via the edge cprov:wasImplicitlyCalledBy) for creating a back-up of the resource. The 'compress' process archives the resource created by the process 'copy' (denoted by the edge 'cprov:wasRecurrentlyCalledBy').

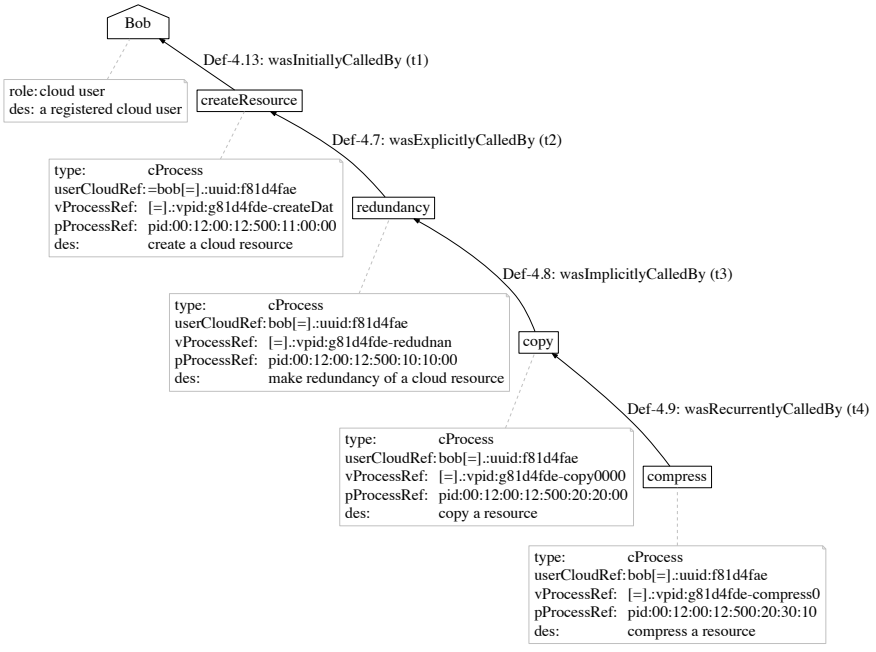


FIGURE 4.3: Traceability Graph - Service Operations

For each process, we can see a list of properties detailing the userCloud reference, virtual reference, and its physical location. Note, these details are simulated for illustration purposes. Each edge contains a value starting with 't' in a bracket. This indicates the ordering of the edge being invoked. For example the cprov:wasInitiallyCalledBy (t1) was invoked first, followed by cprov:wasExplicitlyCalledBy (t2) next.

4.3.2 Service Operation Locality

The second requirement (Req 4.2) models the locality information of the operations. This is facilitated by the 'cprov:Transition' node and 'cprov:hadTransitionState' edge.

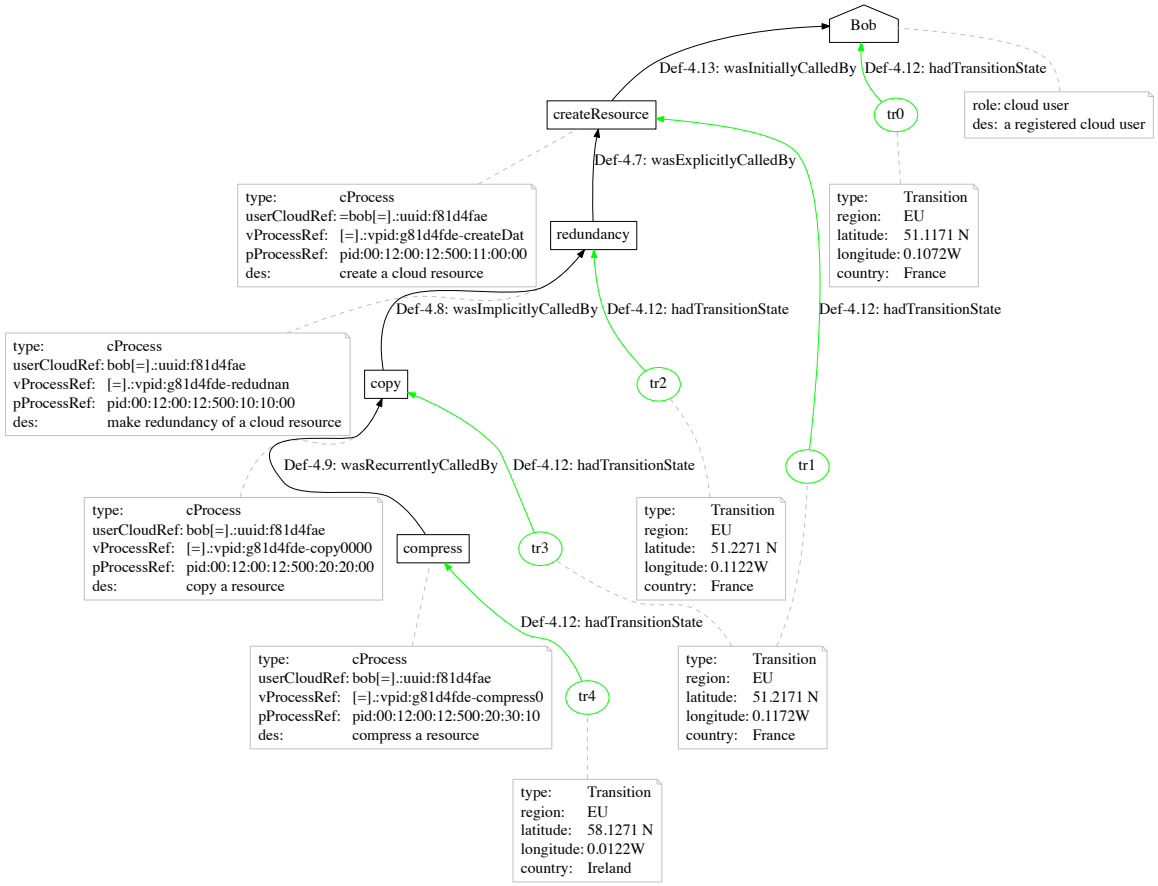


FIGURE 4.4: Traceability Graph - Service Operations Locality

Figure 4.4 shows the added locality information to the process. The 'tr0' element has a relation (cprov:hadTransitionState, see Def 4.12) with the element 'agent' which captures all the geo-location details, such as: region, latitude, longitude and others. The 'tr1' element also has a relation (cprov:hadTransitionState) with the process 'createResource' and 'tr2' with the 'redundancy' process which also captures similar details. With this additional information, we are able to analyse the nature of the operations involved and where it took place.

Since a Transition element contains state and location information belonging to an entity, activity or agent, it is modelled as a separate entity as opposed to being integrated with an activity/entity/agent. This is because the cloud is a highly dynamic environment, where the locality information is likely to change over time. Such change can be reflected on the traceability graph by having a separate cprov:Transition entity linked via the cprov:hadTransitionState. Transition elements are related to each other via the same eventId. Section 4.3.4 presents the event related information for the Transitions. The transition elements identifier (tr1, tr2 .. trN) signifies the ordering of their occurrence.

4.3.3 Service Resource

The third requirement (Req 4.3) requires capturing of the resources within a service. This is facilitated by the 'cprov:cResource', 'cprov:pResource' nodes and the 'cprov:wasVirtualizedBy', 'cprov:wasRepresentationOf, 'cprov:wasReferenceOf' and 'cprov:hadOwnership' edges.

Figure 4.5 shows the addition of the resources. Process 'createResource' uses the cprov:pResource (originDoc) which resides outside of the cloud and generates a new cprov:cResource (newDoc) that belongs to the agent 'Bob'; denoted by the relation 'cprov:hadOwnership'). This resource (newDoc) is used by the 'redundancy' process to produce a back-up resource called 'newDocCopy' (defined by the relations 'cprov:wasRepresentationOf' and 'cprov:wasVirtualizedBy'. The 'compress' process is notified by the 'copy' process to create a compress version of the resource 'newDocCopy-Compress' from the resource 'newDocCopy'.

4.3.4 Service Resource Locality and Grouping

The fourth and fifth requirements (Req 4.4, Req 4.5) require locality information of the resources, and the grouping of resources, processes and agents. Figure 4.6 exhibits location information related to the resources via the 'cprov:Transition' node and 'cprov:hadTransitionState' edge.

The grouping of the resources, agent and operations which belongs to an event is defined by using the 'cprov:event' property of the 'cprov:Transition' element. This is particularly useful when there is a large traceability graph, and only interested in a certain part or certain information in a graph. A part of a graph can be extracted by using the event Id.

From the graph (Figure 4.6) it can be seen the grouping of nodes and their associated edges is achieved by using the same event Id (ev001) of the transition elements. In other words, the traceability graph represents an event containing a group of nodes with their

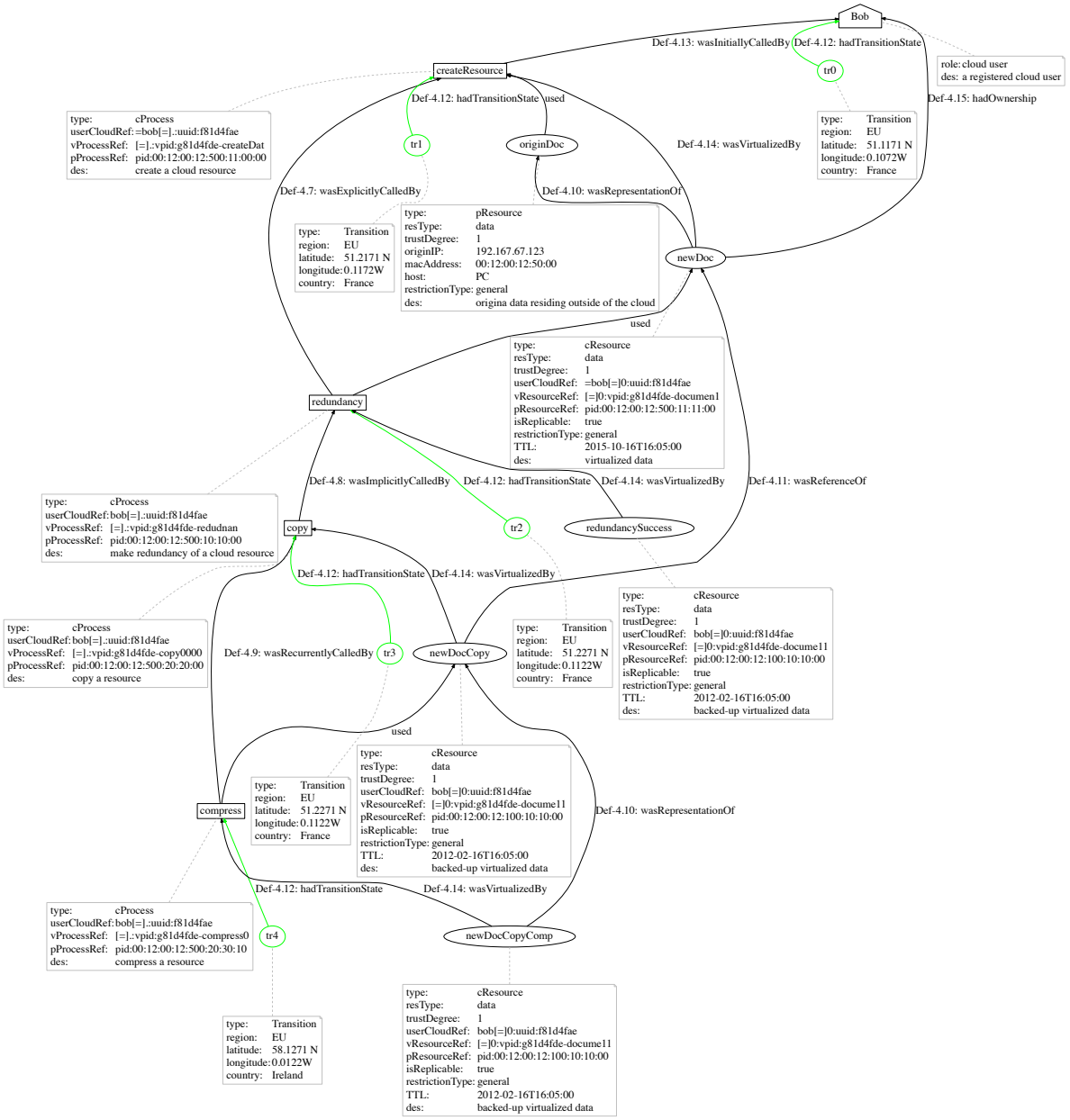


FIGURE 4.5: Traceability Graph - Service Resources

associated edges. If multiple executions of this event occur, we threat each occurrence as a new event (contains different eventId), hence, a new group for each execution.

The four graphs have successfully modelled the five traceability properties, and are able to visualize the complete traceability of an event in the cloud. This enables us to see the ordering of the executions occurred and their dependencies. Such information is vital for commercial services to ensure any service level agreements with the client can be validated using this data. However, the model itself does not define the mechanism(s) to carry out such tasks, but we address this in later chapters.

In this simple example, we have used all the derived nodes and edges defined in the

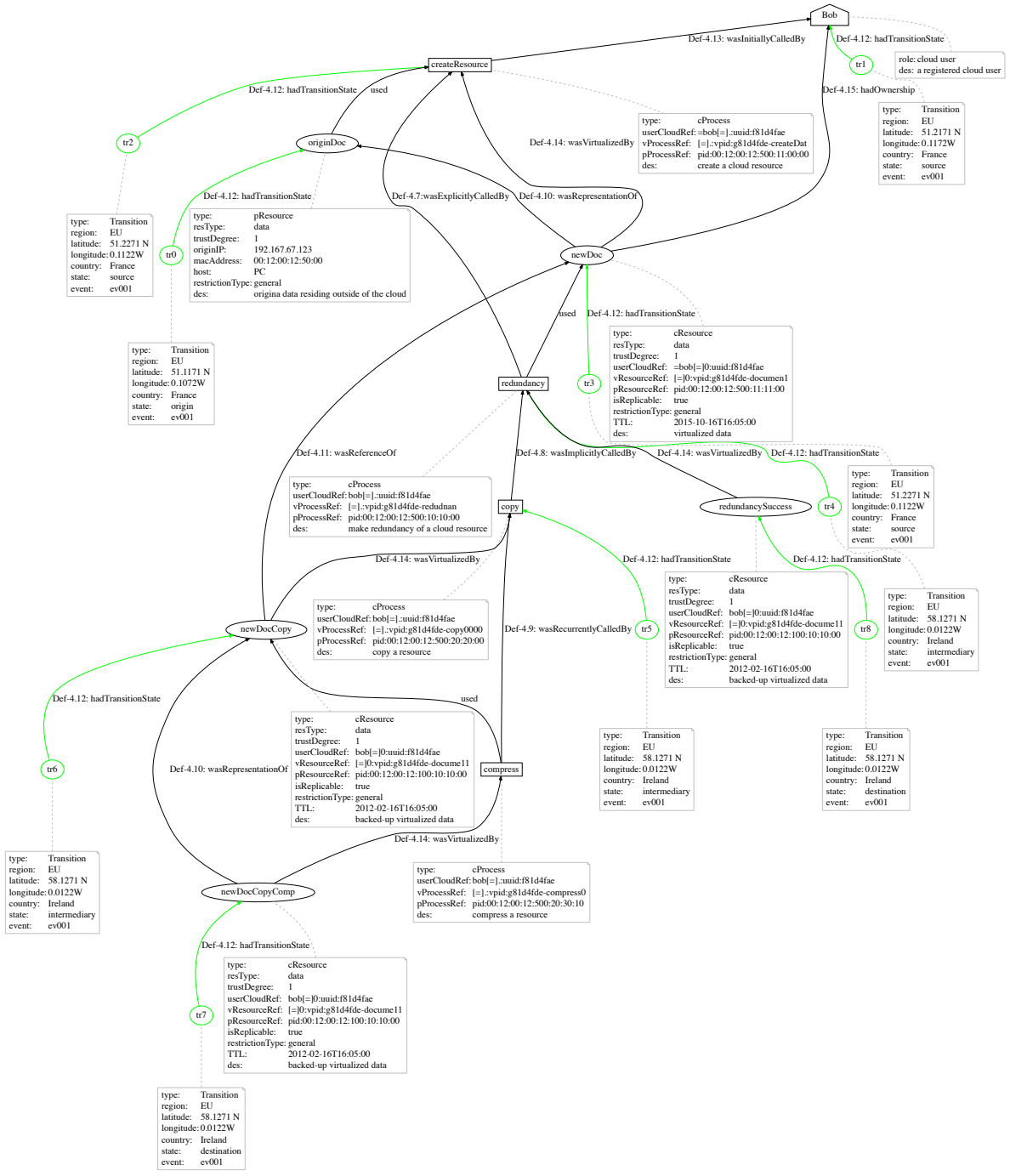


FIGURE 4.6: Traceability Graph - Service Resource Locality & Grouping

cProv traceability model. The cProv model is sufficient for capturing all the traceability requirements of the ConfidenShare service (see Section 2.1).

4.4 Discussion

The cProv traceability model presented in this chapter is able to conceptualize the running of a cloud service by capturing all the interactions and operations that have taken place. It inherits from the base elements and relations of the Prov model, and defines its own specialized subsets.

The model has sufficient level of specialization to comprehensively capture the traceability of a cloud service, which is vitally important for commercial-based services such as the ConfidenShare. Traceability data generated by the model has various usages: diagnostic purpose, understanding user's behaviour and interactions with a service, order of events occurrence, as well as for general service improvements. Moreover, using advanced data mining techniques, it can be used to identify anomalies, threats and other correlations that exist within the traceability dataset.

The model is designed to capture traceability information of cloud based services. It introduces a number of nodes and edges that contains various properties catered for cloud related information. This includes virtual and physical address of running processes, storage of resource, and location information. The model, however, does not define how this information can be extracted from a running cloud environment. This is done on purpose, to make the model implementation independent. It is left to the cloud service supplier to provide their mechanism for extracting such information.

The traceability model is designed to be applied at the service level. The users of the service would generate their own footprint of the traceability data. This data can be used by the diagnostic services to improve user experience or address issues and problems the users have encountered, as well as for monitoring purpose. The visualization of the traceability data at present is of graph structure which, cannot be understood easily by normal users. Extensive research is required in improving presentation and understanding of traceability data in order to be consumed by various user groups: technical/non-technical users, managers and stakeholders.

While comprehensive collection of traceability data can bear many fruits, it also brings a number of challenges: the likes of cost, scalability, quality and performance. The more comprehensive traceability modelling of a service, the larger the size of the data collection (can be more than the service data itself). With the increase in size, this can add additional overhead costs both in terms of resources and monetarily. This can have an impact on the commercial viability and adaptation of the traceability modelling. This raises a fundamental question of, *how comprehensive the traceability collection should be, and the duration it needs to be kept for?*

The comprehensiveness has a direct correlation with the size, and richness of data. The richness of the data is vital for the feature extraction, and in-depth analysis; hence the more comprehensive it is the better the analysis, but this may come with a greater cost,

scalability and large data size. In regards to our service, the choice of comprehensiveness versus partial traceability modelling is primarily determined by fulfilment of business requirements. Due to the domain specificity of the cProvl model, the business requirements can be captured sufficiently with a partial traceability model of the service. This ensures that the collection of data generated is relevant and concise, however over time the continuous growth of data may likely to be unmanageable. It would therefore be necessary to apply various optimization techniques (such as pruning, archiving irrelevant data and others) to manage the growth of data.

4.5 Summary

In this chapter we have proposed a new traceability model cProv by extending the Prov standardized model, and have introduced a number of new nodes and edges to support the traceability requirements. These requirements were validated by modelling samples of traceability graphs. The specialization of the model enables capturing of all the necessary information to have a comprehensive traceability view of the service. The cProv model, it must be stressed, does not introduce new notation to maintain conformity with Prov. Yet at the same time cProv can be extended further to incorporate new requirements and future extensions.

Chapter 5

cProvl - A Policy Language for the Traceability Model

In the previous chapter, we have proposed a traceability model (cProv) based on Prov, and have demonstrated conceptualization of traceability for cloud-based services. This chapter extends this work by proposing a traceability-based policy language. The language has rules and restrictions for a service to be expressed, which are then validated against traceability data.

Many of the existing policy languages [195], [101] used for access control to resources are primarily based on policy expressions, these expressions capture the business requirements, and primarily contain values that do not cater for traceability (provenance) data. The proposed policy language handles both predefined values, as well as dynamically generated values through variables, which are determined at run-time and are validated against traceability data. Such features are needed in fulfilling our service requirements (see Section 2.2).

The contribution of this chapter is as follows: *Describes an abstract syntax for a provenance-based policy language (cProvl) that facilitates declaration of rules and restrictions for a service, based on the Prov vocabulary. Beyond declaring policies with predefined values, it can handle dynamic values via variable referencing, which is determined at run-time. The language runtime retains the provenance of its policy execution, which can be used for auditing purposes.*

5.1 Motivation for a Policy Language

Traceability data contains historical information related to the running of a service. This information provides a chronological order of events that have occurred and where they took place. Such information is vital in understanding if the correct procedures were

followed to obtain the necessary results. However, obtaining such information manually can be fairly complex and time consuming, especially if the traceability store is relatively large. Moreover, if some irregularities or violations are found, it is not so easy to take necessary actions manually. This can be a major problem for services where immediate actions may be necessary.

To mitigate these issues, a policy language is needed that can support the following:

Requirement 5.1 : Declaration of Policy, Request & Response

The capturing of the business logic of a service and its constraints are required to be modelled by policies. A policy needs to be invoked by a request, and associated responses have to be generated. All three (policy, request and response) need to adhere to the same semantics, whereby a policy can use the values of a request and generate a response that can be interpreted by the request generator. A rule containing fine-grained constraints can consist of either singular or compound statements.

Requirement 5.2 : Traceability Data Usage

An execution of a policy has to query traceability data in determining the outcome of a request. To satisfy the interoperability and expressibility, both policy and traceability data need to have the same structure, and be able to examine the traceability at various levels of granularity.

Requirement 5.3 : Policy Enforcement

The outcome of a policy execution determines the type of enforcement needed to be performed by the service. This would be in the form of granted, denied or undetermined.

Requirement 5.4 : Retain Execution Traceability

The running of a policy statement may change the course of execution of a service, such information is important in keeping a record of which decisions were made.

With respect to all these requirements, a new policy language, that is able to fully support traceability data and can model the complex business requirements is required.

5.2 cProvl - Policy Language

From our literature review, we have explored a number of policy languages that operate on data which are non-provenance related [195], [101] and few new languages [204], [65], [184] which are designed to work on provenance data. These languages have limitations in terms of their usage of the provenance graph nodes, edges, associated properties, and their compatibility with Prov.

Now we present the cProvl policy language, which is designed to operate fully with traceability data and can express policies for declaring business constraints (service policies requirement, see Sections 2.2.1, 2.2.2, 2.2.3).

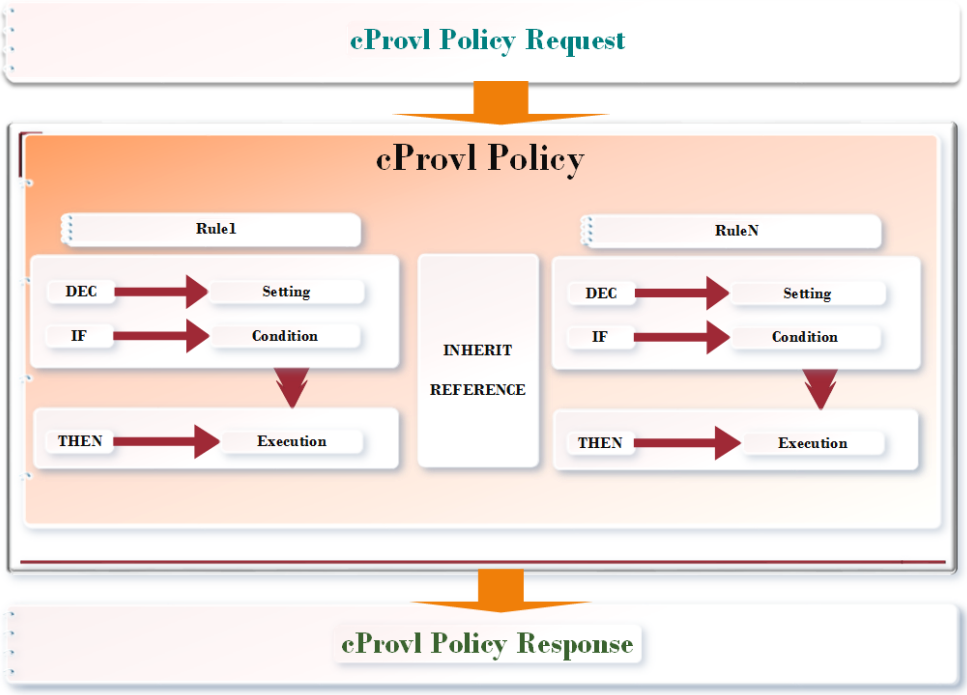


FIGURE 5.1: cProvl Policy Language Structure

Figure 5.1 shows a high-level view of the cProvl policy language, which contains three parts: policy, policy request and policy response. The policy request is designed for constructing service requests (e.g. a request to share a document in the ConfidenShare service). A policy defines the business constraints, which are modelled as rules and is validated against the traceability data that exists at the time to determine the outcome. The outcome is constructed as a policy response, which is acted upon by the service. We can concretely define the syntax of the policy language.

5.2.1 Policy Language Syntax

A list of syntactic constructs and capabilities for the policies, request and response are defined in Table 5.1.

Syntax for Policy	
POLICY ::=	“POLICY-BEGIN” provEntity “ENV” provEntity RULE+ “POLICY-END”
RULE ::=	“RULE” provEntity ruleOperator? “DEF” provEntity defStatement “IF” (targetQuantifier targetIDs*)? “SUCH THAT” condition* “THEN” permission
ruleOperator ::=	“INHERIT” provEntity |“REFERENCE” provEntity
defStatement ::=	“cprovd:all” |“cprovd:event” |“cprovd:node” |“cprovd:edge”
targetQuantifier ::=	“FOR ALL” |“THERE EXIST”
targetIDs ::=	{ (“ex:” identifier)* |(variable “r:” identifier)* }
variable ::=	“new” |“s-ref” |“d-ref”
identifier ::=	letter (letter |digit)*
condition::=	{Grouping}* {conditionalStatement}
conditionalStatement ::=	provStatement {booleanOperator |conditionalOperator}
provStatement ::=	“entity” |“activity” |“agent” |“used” |“wasDerivedFrom” |“wasAssociatedWith” |“wasInformedBy” |“wasGeneratedBy” |“wasAttributedTo”
provEntity::=	“entity (ex:” identifier “, [” (property)* “])”
property::=	(“cprovd:” identifier “=” value)
value ::=	letter (letter |digit)*
condition::=	{Grouping}* {conditionalStatement}
conditionalStatement ::=	provStatement {booleanOperator |conditionalOperator}
provStatement ::=	“entity” |“activity” |“agent” |“used” |“wasDerivedFrom” |“wasAssociatedWith” |“wasInformedBy” |“wasGeneratedBy” |“wasAttributedTo”
provEntity::=	“entity (ex:” identifier “, [” (property)* “])”
property::=	(“cprovd:” identifier “=” identifier)
Grouping ::=	{Grouping (conditionalStatement+ {booleanOperator |conditionalOperator}}
booleanOperator ::=	“&& ” |“||” |“!”
conditionalOperator ::=	“== ” |“<” |“>” |“=>” |“<= ”
permission ::=	“cprovd:permit” |“cprovd:deny” |“cprovd:indeterminate” |“cprovd:not-applicable”
POLICY-REQUEST::=	“REQUEST-BEGIN” provEntity provStatement+ “REQUEST-END”
POLICY-RESPONSE::=	“RESPONSE-BEGIN” provEntity provStatement+ “RESPONSE-END”

Table 5.1: cProvl Policy Language Syntax

The cProvl's grammar for defining the structure of a policy, request and response has been presented as Backus-Naur-Form [107]. The syntax for the language is defined by reserved words (upper-case), and statements are expressed using the Prov notations.

Each statement can contain one or more properties that are used to define language specific functionalities, which are used within different parts of a policy (rule, target, condition, and others) to enrich the vocabulary. These properties are summarized in the Table 5.2 below.

cProvl properties	
cprovl:actionType	the type of actions permitted are: cprovd:permit, cprovd:deny, cprovd:indeterminate, cprovd:not-applicable and cprovd:obligation
cprovl:description	provides detailed description
cprovl:categoryType	defines request fields: cprovd:subject, cprovd:resource, cprovd:action and cprovd:environment
cprovl:environmentName	unique identifier of the request handler
cprovl:msg	provides related information
cprovl:obligationAction	nine actions are: cprovd:record, cprovd:delete, cprovd:copy, cprovd:read-only, cprovd:read-write, cprovd:check-user-history, cprovd:non-share, cprovd:block-user and cprovd:remove-user
cprovl:obligationType	there are two obligation types: cprovd:alert and cprovd:notify
cprovl:operationName	id of the operation that require access
cprovl:order	defines the ordering of rules, it can be either cprovd:before or cprovd:after
cprovl:part	override parts are: cprovd:target, cprovd:condition and cprovd:execution
cprovl:reqAgent	id of the request agent for the input request
cprovl:resAgent	id of the response agent
cprovl:serviceName	id of the service for the request
cprovl:resourceName	id of the resource that require access
cprovl:range	defines the range of target. It can take any of the following values: cprovd:all, cprovd:event, cprovd:node and cprovd:edge

TABLE 5.2: cProvl Policy Language Properties

Various namespaces (on top of those already defined in Table 4.1) have been used by the policy language to qualify names belonging to elements and their properties (see Table 5.3).

Policy Language Namespaces		
prefix	**suffix**	**description**
cprovl	http://labs.orange.com/uk/cprovl#	Used by the cProvl elements and its attributes
r	http://labs.orange.com/uk/r#	Used to qualify variables

TABLE 5.3: cProvl Policy Language Namespaces

The following sections explain more in details the cProvl policy language syntax (defined in Table 5.1).

5.2.2 Policy Structure

A cProvl policy needs to adhere to the following structure.

```
1  POLICY-BEGIN ()
2      //declaration of a policy
3      entity(ex:policyId, [cprovl:description=""])
4      ENV
5          entity(ex:envId, [cprovl:environment=""])
6      RULE
7          //declaration of a rule
8          entity(ex:ruleId, [cprovl:description=""])
9             //   - -  conditions - - -
10     RULE
11         entity(ex:ruleId, [cProvl:description=""])
12        //  - -  conditions - -
13 POLICY-END
```

The start of a policy is declared with the keyword 'POLICY-BEGIN' and ends with the 'POLICY-END'; anything that falls outside of these clauses results in a syntax error. The language specific attributes are defined using the namespace prefix cprovl. A unique identification of a policy is declared with an entity that contains a unique Id reference (ex:policyId), followed by a description attribute cprovl:description. This is followed by rule declarations (see below Section 5.2.3).

A service can have multiple policies which encompasses various business logics. When a policy engine receives a policy request from a service user, it needs to map the request to the correct policy(s). This is handled by the 'ENV' section. An environment identifier (cprovl:environment) is used to indicates a policy related to particular business logic. It is possible for multiple policies to have the same identifier. The environment identifier needs to be included in the policy request (see Section 5.2.4 for more details) for the policy engine to select the correct policy(s).

An example.

```
POLICY-BEGIN ()
    //declaration of a policy
    entity(ex:policyId, [cprovl:description=""])
    ENV
        entity(ex:envId, [cprovl:environment='cprovd:reg.share.confidenshare.
         labs.orange.com'])
        ........
POLICY-END
```

This example shows registration related business logic is handled by this policy. A client sending a registration related request will be responded to by this policy.

A policy can have one or more rules, which are declared using the keyword 'RULE'.

5.2.3 Rule Structure

A rule is designed to execute certain business logic for controlling resources. A rule has the following structure:

```
RULE
    entity(ex:ruleId, [cprovl:description=""])
    [INHERIT || REFERENCE]
    //rule constraints
    DEF
        entity(ex:scope, [cprovl:range=""])
        //scope declaration
    IF  [THERE EXIST || FOR ALL] SUCH THAT
        [CONDITIONS]
    THEN
        [EXECUTION]
```

Each rule is identified by an entity that contains a unique identifier Id, followed by a brief description of the rule. This is followed by the optional rule operators.

5.2.3.1 Rule Operator

A policy consists of one or more rules, which may contain dependencies between them. Dependencies can be of various forms. For example, a rule may be required to be executed before another or one rule may be used by another. Two operators have been defined to handle such functionalities.

- INHERIT - This operation allows the inheritance of a rule and overriding of an inherited rule. It is possible to override the target, condition and execution sections of the inherited rule by the base rule.
- REFERENCE - This defines referencing of other rules, which can be executed before or after executing the current rule.

```
1 RULE
2     entity(ex:ruleId, [cprovl:description=""])

3     INHERIT
4         entity(ex:policyId, [cprovl:rule='ex:ruleId1',
5         cprovl:part="cprovd:target | cprovd:condition | cprovd:execution"])
6     DEF
```

INHERIT is an operator that is required when one rule needs to inherit from another. It is *not* possible to have multiple inheritance, only one rule can be inherited from. This is to minimize the possible conflicts in the inheritance chain (also known as the deadly diamond of death [128]). The purpose of the inheritance is to reuse a part or whole of an existing rule by a new rule. In an organization it is normal practise to update, revise or extend existing policies. The inheritance operator can be used to override part of the rule (using the'cprovl:part' attribute) that needs amending. The above shows a rule inheriting from another rule (see lines 4-5) and can choose to override the target, condition or execution part of the inherited rule using cprovl:part variable.

The REFERENCE operator defines the ordering of execution of rules, where more than one rule is used to capture service logic.

```
1 RULE
2     entity(ex:ruleId01, [cprovl:description=""])

3     REFERENCE
4         entity(policyId01, [cprovl:rule = 'ex:ruleId02',
5         cprovl:order = 'cprovd:before' | 'cprovd:after'])

6     DEF
```

The ordering of rules is declared using the cprovl:order property. It allows a rule (ex:ruleId01) to be executed either before (cprovd:before) or after (cprovd:after) another rule (ex:ruleId02). In order to avoid rule cycles, i.e ex:ruleId01 calling ex:ruleId02 and ex:ruleId02 calling ex:ruleId01, only unidirectional calls are allowed. In other words, if (ex:ruleId01) calls (ex:ruleId02) and (ex:ruleId02) calls (ex:ruleId03), then (ex:ruleId03) is not allowed to call (ex:ruleId02) or (ex:ruleId01) and (ex:ruleId02) is not allowed to call (ex:ruleId01).

This is followed by the declaration of the scope.

5.2.3.2 Scope

The scope allows users to be selective in defining the range of data (structure) to be used by a policy. It is declared under the keyword 'DEF'. A scope is set by using the cprovl:range attribute.

An example:

```
1 DEF
2     entity(ex:scopeId, [cprovl:range='cprovd:all'])
```

This attribute (cprovl:range) can take a range of values based on the nature of the granularity. The levels of filtering can be defined as coarse-grained or fine-grained. Coarse-gained looks at the overall data or an event. Fine-grained looks at elements at the node or edge level.

- cprovd:all - match all the provenance entries (coarse granularity)
- cprovd:event - scope at the event level (coarse granularity)
- cprovd:node - entity, activity and agent (fine granularity)
- cprovd:edge - wasGeneratedBy and others (fine granularity)

Based on the service requirements, a policy can choose to validate its statements against the various levels of granularity.

The scope is followed by the target section.

5.2.3.3 Target

Quantifiers have been used in many systems [125], [124]. A target defines one or more IDs for a rule, which is matched using existential/universal quantifications.

- FOR ALL - match all occurrence of the criteria
- THERE EXIST - match at least one occurrence of the criteria

We can express the two quantifiers as follows:

```
IF THERE EXIST (IDs | variables) SUCH THAT [CONDITIONS]

IF FOR ALL (IDs | variables) SUCH THAT [CONDITIONS]
```

The ID can either be denoted as an entity, agent or activity. There can be multiple IDs in one statement delimited by the commas. An ID is matched against the traceability store. If more than one ID exists, all of them must match in order for a rule to validate the conditional statements (see Section 5.2.3.5), otherwise the rule check is failed.

An example:

```
IF THERE EXIST (ex:meetingRequest) SUCH THAT
    [CONDITIONS]
```

The above expression can be read as follows: match the ID ex:meetingRequest with an identifier in the traceability store. Note that the ex:meetingRequest ID is a static reference and cannot change. However, there may be cases where dynamic values are required, for example, any generated files. This can be expressed by using variables.

5.2.3.4 Variable

A variable allows dynamic values to be bound at execution time, either for an execution of a statement or a dynamic service request. In order to distinguish a variable from a content ID, they are declared with the reserved namespace denoted by the prefix 'r'.

```
    [new | s-ref | d-ref] [r:varName]
```

Variables may denote values of arbitrary data type and do not require any explicit typing. The actual data type is determined dynamically. Variables can be of two types: 'new' and 'ref'.

- **new -** keyword is used to define a *dynamic variable*. The value of the variable is populated by the outcome of an expression, this can be singular or multiple values in the condition section of a rule (see Section 5.2.3.5). Once declared, the variable can be used in multiple places within a rule (the *content* and *type* is determined at runtime). Once the value is bound for a variable, it is retained for the duration of the policy execution.

```
1 IF  THERE EXIST (new r:req) SUCH THAT
2       wasGeneratedBy (r:req, ex:session)
```

This expression declares a dynamic variable called r:req. The value is populated when the wasGeneratedBy edge is matched against the traceability store. The generated value of variable r:req can now be used by other statements.

For example:

```
1 IF  THERE EXIST (new r:req) SUCH THAT
2       wasGeneratedBy (r:req, ex:session)
3       used (ex:meeting, r:req)
```

In this example, the populated value of the r:req (one or more entities) will be used for validating the statement "used". In the case of multiple values, if there is at least one match found against the traceability store, then the match is successful.

- **ref -** keyword expresses a *reference variable*. It is used to obtain a reference from an existing object, typically from an input request. Its value is bound by the execution of the *targets of a rule*.

  There can be of two types:

  - **s-ref** : Refers to a variable that holds values from the request category (such as the resource, action or subject (see Section 5.2.4)) of the service request. These values are used by the conditional statements.
  - **d-ref** : It is similar to s-ref, but the variable values are also matched against the traceability store to check for their existence. If all matches are found, the values are made available to the conditional statements, otherwise the overall rule validation is considered failed.

```
1    IF  THERE EXIST (d-ref r:userA) SUCH THAT
2        hadOwnership(ex:e001, r:userA)
```

This expression checks if the requested user (r:userA) has an ownership to entity ex:e001. The r:userA values are populated from the request (e.g. ex:Bob) and is used by the edge hadOwnership to match against the traceability store.

An example of use of both dynamic and reference variables.

```
1 IF  THERE EXIST (new r:resource, s-ref r:userAgent) SUCH THAT
2       wasGeneratedBy (r:resource, ex:session)
3       wasAssociatedWith(ex:session, r:userAgent)
```

The above example shows use of dynamic and reference variables in one statement. The variable 'userAgent' is used to reference an agent from the request, and r:resource is a reference to an entity in the traceability store. Both attribute values are determined at run-time.

If the validation of the all targets are successful (note the dynamic variables are implicitly considered to be true, since the values are obtained at the validation of conditional statements), then the conditional statements are validated. The next section explores the conditional statement in more detail.

5.2.3.5 Conditional Statements

The language supports conditional expressions on traceability data. The conditional statements are directly based on the Prov vocabulary, they can be either an entity, activity, agent or a relation. It is possible to have multiple statements in a rule, separated by a logical operator. If none defined, an implicit logical 'and' (&&) is assumed. A validation of a statement against the traceability store would return true if a match is found, otherwise a false is returned.

An example:

```
1 IF ..... SUCH THAT
2    entity (ex:meetingRequest, [prov:type='cprov:cResource'])
3    wasGeneratedBy (id, ex:meetingRequest, ex:meetingSession)
4    wasAttributedTo (id, ex:meetingRequest, ex:matt))
5 THEN
```

In this example, each of the statements is matched against the traceability store. If any of the matched outcome is false, then (due to implicit 'and' between statements) the overall outcome of the statements is false. However, this can be fairly restrictive, and may not be the desired flow of execution. For example, if only one of three statements are required to be true, and one of them appears to be false, then all the statements will be considered false. For the correct flow of execution, explicit logical operators are required.

5.2.3.6 Logical Operators

The logical operators can be used in between statements to provide additional condition capabilities, these are:

```
1   && (and),    || (or),    ! (not)
```

An example:

```
1 !  wasGeneratedBy (ex:meetingRequest, ex:mSession)
2 && wasAssociatedWith(ex:mSession, ex:matt)
3 || wasAssociatedWith(ex:session, ex:john)
```

In this example, the first statement (wasGeneratedBy) is matched against the traceability store. If the match is true, it would be negated to false, and vice-versa.

If a statement contains a variable and the variable is not bound to any value, the value would be bound by matching the statement with the traceability store. If successful, the result of the match (true) would be negated (false). However, if the variable already bound to values, then the statement would be matched against the store with the values from the variable and the outcome (true or false) would be negated.

The result of the second statement computes a logically AND with the first statement and if it evaluates to false the remaining statements are ignored. However, if the intent was to evaluate the result of the first statement with the second or third, then this may produce an incorrect outcome. In such cases, groupings are required.

5.2.3.7 Grouping

Grouping of conditional statements allow them to be combined and matched collectively in a certain order, starting with inner most grouping being matched first, followed by the next outer and so on.

An example:

```
wasGeneratedBy  (ex:meetingRequest, ex:session)
{
&&
    {
       wasAssociatedWith(ex:session, ex:matt)
       || wasAssociatedWith(ex:session, ex:john)
    }
}
```

The groupings are facilitated by use of curly brackets { }. It is possible to have nesting groupings. With the addition of curly brackets, it is now possible to construct more complex conditional statements. The inner curly braces are matched first followed by the outer ones.

Logical operators may not be expressive enough if there is a need for specific attribute values within a statement to be matched. In such cases, conditional operators are required.

5.2.3.8 Conditional Operators

By default, a statement which contains static values has an implicit equal operator. During validation, all the fields in a statement are matched against the traceability store.

An example:

```
//declaraing a document with a trust degree of 0.7
entity(ex:resId, [prov:type='cprov:cResource', cprov:type = 'cprovd:data',
 cprov:trustDegree = "0.7", -,-,-, -, cprov:TTL = "2014-11-16T16-05-00"])
```

In this example, if all the attributes values match, then the statement is valid, otherwise it is invalid.

However, if we wanted to express other operations on the statement's attribute values, then explicit arithmetic operators are required, this can be used in conjunction with the logical operators.

```
  == (equal),  < (less than), > (greater than),  >= (greater than or equal),
  <= (less than or equal)  != (not equal)
```

For example, to check if an entity statement's attribute ('cprov:trustDegree') value is greater than a certain threshold value (0.5). This can be expressed as follows:

```
1 //declaraing a document with a trust degree of 0.7
2 entity(ex:resId, [prov:type='cprov:cResource', cprov:type = 'cprovd:data',
3   cprov:trustDegree = "0.7",  -,-,-, -, cprov:TTL="2014-11-16T16-05-00"])
4                                   >
5 //composing a new entity  with a trust value of 0.5 to be compared
6 entity(ex:ob1, [cprov:trustDegree="0.5"])
```

In this example, the threshold value (defined by the attribute cprov:trustDegree) is declared by composing an entity statement (ex:ob1) (see line 6). The greater than notation in line 5 indicates both statements (ex:resId and ex:ob1) attributes value needs to be compared for inequality. The attribute names defined in both entities must match in order for them to be selected for inequality check. In this example, it is checking if 0.7 is greater than 0.5, which it is and the outcome is true. The advantage of this approach is that it can handle comparisons of multiple property values simultaneously.

The validation of all conditional statements determines the course of action to be taken. This is defined in execution statements.

5.2.3.9 Execution

The execution statements of the rule presents the outcome (permission) of a valid rule (based on the satisfactory match of the target and conditional statements). The permission (outcome) is expressed by the following entity.

```
1 entity(ex:result, [prov:type='cProv:cResource', cprovl:actionType='',
2     cprovl:resourceId='', cprovl:msg=''])
```

The actionType is a mandatory field, which defines the nature of the outcome. The following statements are valid outcomes:

- cprovd:permit - grant permission
- cprovd:deny - deny permission
- cprovd:indeterminate - cannot determine
- cprovd:not-applicable - irrelevant request

These actions are coarse-grained since they apply to all resources. For fine-grained access, a particular resource can be specified explicitly using the resourceId attribute.

Sometimes there may be a need to carry out post actions based on the nature of the outcome. For example, if an outcome was cprovd:deny, an alert may need to be raised to review the user's account. In such cases we need obligation.

Obligation - The outcome may also require additional actions or controls to be performed after the response sent to the user. This is defined by an Obligation.

Obligation can be defined as follows:

```
entity(ex:result,[prov:type='cprov:cResource', cprovl:actionType='', cprovl:resourceId='',
 cprovl:obligationType='', cprovl:obligationAction='', cprovl:msg=''])
```

- **obligationType** - 'cprovd:alert' , 'cprovd:notify'.
- **obligationAction** - 'cprovd:record', 'cprovd:delete', 'cprovd:copy', 'cprovd:read-only', 'cprovd:read-write', 'cprovd:check-user-history', 'cprovd:non-share', 'cprovd:block-user' or 'cprovd:remove-user'.

There are two obligation types (cprovl:obligationType): cprovd:alert and cprovd:notify. The cprovd:alert is used to imply some urgent action is required. The cprovd:notify, on the other hand, is used to express complementary or informative actions. The nature of the action is defined by the attribute cprovl:obligationAction.

In addition to defining outcome (with or without obligation), the outcome has an association, which is represented by the relation 'wasAttributedTo'.

An example of a response without obligation shown below:

```
THEN
    entity (ex:result, [prov:type='cprov:cResource',cprovl:actionType='cprovd:deny',
     cprovl:resourceId='ex:mData', cprovl:msg=""])

    wasAttributedTo (ex:result, r:ag1, [prov:type='cprov:hadOwnership',
     cprov:ownershipType='cprovd:possession'])
```

This example shows the result (ex:result) entity is used to model the outcome of the rule. The cprovl:actionType attribute defines the outcome (cprovd:deny) for the resource (cprovl:resourceId) ex:mData. The result is attributed to the user (ag1). In other words, it denies (cprovd:deny) agent (ag1) access to the resource ex:mData. However, it does

not define what should be done as a result of the deny outcome. This can be expressed with the cprovl:obligationType and cprovl:obligationAction attributes.

An example of response with obligation is as follows.

```
THEN
    entity (ex:result,[prov:type='cprov:cResource',cprovl:actionType='cprovd:deny',
     cprovl:resourceId='ex:mData',  cprovl:obligationType='cprovd:alert',
     cprovl:obligationAction= 'cprovd:record', cprovl:msg=""])

    wasAttributedTo ( ex:result, r:ag1, [prov:type='cprov:hadOwnership',
     cprov:ownershipType='cprovd:possession'])
```

The above example denies (cprovl:actionType) agent (ag1) access to the resource ex:mData and raises an obligation as an alert to record the event.

We have now defined the structure of a policy. However, a policy does not operate on its own, it needs to be invoked by a policy request, and based on the policy outcome, a response needs to be generated.

5.2.4 cProvl Policy Request Structure

Multiple requests from various users can be sent to the policy engine to be authorized by one or more policies. Such requests need to be composed in a structured way so that they can be processed by the policy execution engine.

A cProvl policy request is divided into four parts (also referred to as categories): subject, resource, action and environment.

subject - A 'subject' is defined as the user (request originated by) of the request. This is modelled as an agent.

resource - A 'resource' refers to a content, such as file or a piece of data the requester is interested in. This is expressed as an entity.

action - An 'action' defines an operation, such as 'share-file', 'delete-user' and others. Since only the name of the action (cProcess) is required, we have decided to model it as an entity.

environment - An 'environment' defines an reference that is used by the policy engine to map to a policy. It is expressed as an entity.

Figure 5.2 shows the cProvl policy request structure.

```
1  REQUEST-START
2      entity(ex:requestId01,[cprovl:description=""])
3      REQUEST-DEC

4        // main body of the request
5        agent(ex:agentId01, [cprovl:categoryType='cprovd:subject',
6         cprovl:reqAgent="", cprovl:description=""])
7        entity(ex:resId01, [cprovl:categoryType='cprovd:resource',
8         cprovl:resourceName="", cProvl:description=""])
9        entity(ex:oprId01, [cprovl:categoryType ='cprovd:action',
10        cprovl:operationName="", cprovl:description=""])
11       entity(ex:enrId01, [cprovl:categoryType ='cprovd:environment',
12        cprovl:environmentName="", cprovl:description=""])

13       entity(ex:tr1, [prov:type='cprov:Transition', cprov:region='provd:GB',
14        cprov:country='provd:England', cprov:latitude="51.5171 N",
15        cprov:longitude="0.1072W", cprov:state='cprovd:origin', cprov:event='ex:ev001'])

16      //relation edges
17       wasAttributedTo (ex:wat01, ex:requestId01, ex:agentId01, [prov:type=
18         'cprov:hadOwnership', cprov:ownershipType='provd:originator'])
19       wasAttributedTo (ex:wat02, ex:resId01, ex:agentId01, [prov:type=
20         'cprov:hadOwnership', cprov:ownershipType='provd:originator'])
21       wasAttributedTo (ex:wat03, ex:oprId01, ex:agentId01, [prov:type=
22         'cprov:hadOwnership', cprov:ownershipType='provd:originator'])
23       wasAttributedTo(ex:wat04, ex:tr1, ex:agentId01, [prov:type=
24         'cprov:hadPresence', cprov:method=''])
25 REQUEST-END
```

FIGURE 5.2: Policy Request Structure

Figure 5.2 shows the structure of a cProvl policy request, which is used by a service user to compose a request that is handled by a policy engine. The request includes the requested user, required access resource and operation. We can see lines 5-6 define the request subject (cprovl:categoryType attribute) as an agent. Lines 7-12 declare a resource, action and environment as entities. Lines 13-15 declare a transition, which captures the locality information related to the request, and lines 17-24 define edges as attributions to the agent for constructing the request.

5.2.5 cProvl Policy Response Structure

The execution of a policy results in a response, whose structure is illustrated by Figure 5.3. The response structure is used by the policy engine to define the outcome of a policy request. The structure defines the request outcome and process involved.

Line five references the request Id. The policy language is responsible for generating the response which is modelled as an agent (line 7). The processes involved in producing the result (ex:resultId01) are defined in lines 11-14. Lines 19-35 present the relationships that exist between the policy language, processes and entities.

```
RESPONSE-START
    entity(ex:responseId01, [cprovl:description=""]) //response Id

    RESPONSE-DEC
       //request  Id
        entity(ex:requestId01,[prov:type ='cprov:cResource'])
       //policy language
        agent(ex:policyLan01,prov:label="policy language")
        entity(ex:resultId01, [prov:type ='cprov:cResource',
          cprovl:decision='cprovd:permit', cProvi:description=""])

        //processes for target and conditional statements
        activity(ex:targetProc01, -, [prov:type='cprov:cProcess',
          cprov:userCloudRef="", cprov:vProcessRef="", cprov:pProcessRef="")
        activity(ex:conditionProc01,-,[prov:type='cprov:cProcess',
          cprov:userCloudRef="", cprov:vProcessRef="",cprov:pProcessRef="")
        entity(ex:tr1,[prov:type='cprov:Transition',cprov:region='',
          cprov:country='',cprov:latitude="", cprov:longitude="", cprov:state='',
          cprov:event=''])

        // relation edges
        wasDerivedFrom(ex:wdf01, ex:resultId01, ex:e-responseId01, -,-,-,
          [prov:type='cprov:wasReferenceOf', cprov:refType=''])
        wasAttributedTo (ex:watId01, ex:resultId01, ex:policyLan01,
          [prov:type='cprov:hadOwnership', cprov:ownershipType=''])
        wasInformedBy(ex:wib01 , ex:conditionProc01, ex:targetProc01,
          [prov:type='cprov:wasExplicitlyCalledBy', cprov:implicitType='',
           cprov:callComm='', cprov:callMedium='', cprov:callNetwork=''])
        wasGeneratedBy(ex:wvb01, ex:resultId01, ex:targetProc01,-,
          [prov:type='cprov:wasVirtualizedBy',  cprov:purpose=''])
        wasGeneratedBy(ex:wvb02, ex:resultId01, ex:conditionProc01,-,
         [prov:type='cprov:wasVirtualizedBy', cprov:purpose=''])
        wasAssociatedWith(ex:wicb01, ex:targetProc01, ex:policyLan01,
          [prov:type='cprov:wasInitiallyCalledBy', cprov:purpose='',
          cprov:accessMedium='', cprov:accessNetwork=''])
        used(ex:ex:used01, ex:targetProc01, ex:requestId01, -)

        wasAttributedTo(ex:watId02, ex:tr1, ex:policyLan01,
         [prov:type='cprov:hadTransitionState, cprov:method=''])
RESPONSE-END
```

FIGURE 5.3: Policy Response Structure

5.3 Validating Policy Language Requirements

We now demonstrate validation of the four policy language requirements (see Section 5.1) by using one of the policies from the service requirements.

If a file (fileA) is marked as 'confidential', only the originator is allowed to share it another user (userB), re-sharing is not allowed.

5.3.1 Declaration of Policy, Policy Request and Policy Response

The first requirement needs to capture the business logic via a policy, and invocation of the policy using policy request and response generation.

cProvl Policy

```
POLICY-BEGIN
    entity(ex:policy1, [cprovl:description=""])

    ENV
        entity(ex:confidential.share.confidenshare.labs.orange.com, [cprovl:description=""])

    RULE
        entity(ex:rule1, [cprovl:description="confidential sharing"])

        DEF
           entity(ex:scope, [prov:type='cprov:cResource',
             cprovl:range='cprovd:all'])

         IF THERE EXIST (d-ref r:ag001, r:fileA, ex:share) SUCH THAT

            // check if the file is 'confidential'
           entity (r:fileA, [prov:type='cprov:cResource', cprov:restrictionType =
             'cprovd:confidential'])

           // check if the ag001 is the owner of the file
             wasAttributedTo(r:fileA, r:ag001, [prov:type='cprov:hadOwnership',
                  cprov:ownershipType='cprovd:originator'])

         THEN
             //action to be taken
              entity(ex:response, [prov:type='cprov:cResource', cProvl:actionId=
                 'cprovd:permit', cprovl:resourceId='ex:meetingFile'])
              wasAttributedTo (ex:response, r:ag1, [prov:type=
                 'cprov:hadOwnership', cprov:ownershipType='cprovd:possession'])

     //this rule is executed if the first rule is not applicable, it denies all
     RULE
        entity(ex:rule2, [cprovl:description="deny all"])

         IF THERE EXIST () SUCH THAT
         //
         THEN
              entity(ex:response1, [prov:type='cprov:cResource', cProvl:actionId=
                 'cprovd:deny', cprovl:resourceId='r:fileA'])
              wasAttributedTo (ex:response1, r:ag1,[prov:type=
                 'cprov:hadOwnership', cprov:ownershipType='cprovd:possession'])

POLICY-END
```

FIGURE 5.4: Policy

Figure 5.5 shows the traceability entries used by the policy (see Figure 5.4) to match the target and conditional statements. We can see the confidential property of fileA is modelled using the attribute cprov:restrictedType. The 'fileA' is replicated by the 'copy' process to be shared as a copy. The 'notification' process which has a wasRecurrently-

CalledBy edge to the 'share' process waits to be notified then generates a notification response denoted by the 'outcome' entity.

The policy (Figure 5.4) first checks if the file requested by the user is marked as confidential (lines 12-13). If a valid match is found in the traceability store entries (see Figure 5.5), the policy then checks if the requested user is the originator of the file (lines 15-16). Note, the user and file name are defined as variables (line 10), and their values are obtained at run-time from the request. The policy is generic enough to handle requests from various users.

If the first and second conditions are true (ex:rule1) then the sharing of the file is permitted, otherwise denied.

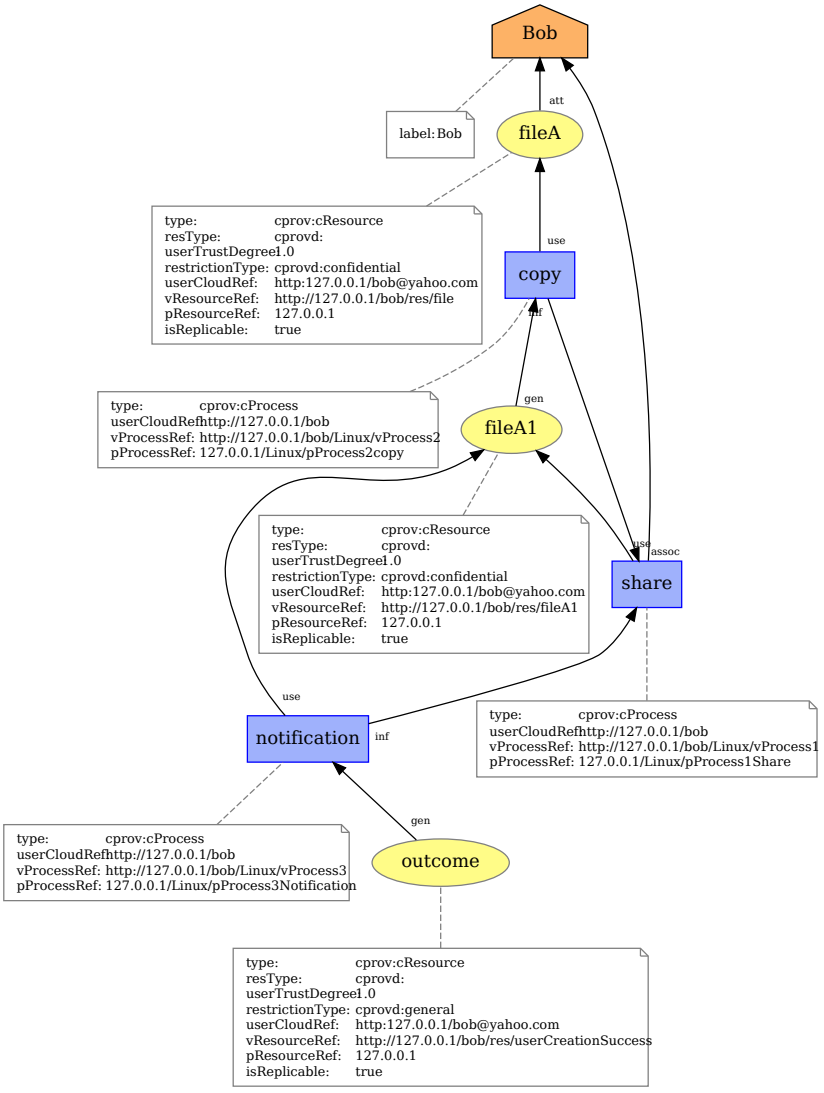


FIGURE 5.5: Traceability Entries

cProvl Policy Request & Response

Figure 5.6 shows the policy request of a share operation for the resource fileA.

```
1  REQUEST-START
2      entity(ex:requestId01,[cprovl:description=""])
3      REQUEST-DEC
4        // main body of the request
5        agent(ex:Bob, [cprovl:categoryType='cprovd:subject',
6          cprovl:reqAgent='ex:Bob'])
7        entity(ex:document1, [cprovl:categoryType='cprovd:resource',
8          cprovl:resourceName='ex:fileA'])
9        entity(ex:share, [cprovl:categoryType ='cprovd:action',
10         cprovl:operationName='ex:share'])
11       ..........
12 REQUEST-END
```

FIGURE 5.6: Policy Request

In this request, lines 5-6 shows the request subject, which is modelled as an agent (ex:Bob). The request resource (lines 7-9) is defined as an entity (ex:fileA) and the action (lines 10-11) ex:share is also defined as an entity.

The response to the policy request is shown in Figure 5.7. The response is generated by tracing the execution steps carried out by the policy engine. Lines 6-7 shows the response outcome (cprovd:permit). Lines 11-32 provides the traceability of the policy execution, which includes the processes and entities that were involved and their associated relationships.

5.3.1.1 Traceability Data Usage and Policy Enforcement

The second requirement needs the usage of traceability data for validating the policies. From the policy (see Figure 5.4) lines 12-16 define a number of traceability statements. Each statement requires validating against a traceability graph (Figure 4.6, an example of traceability graph) in the traceability store to determine the outcome.

The third requirement needs policy enforcement to be acted upon by a service based on the policy validation outcome. The cProvl policy language defines the course of action to be taken based on validating the target and conditional statements. The response (see Figure 5.7) generated by the policy language contains the nature of enforcement should be applied by client (e.g cprovd:permit). The client determines how the enforcement should be performed.

5.3.1.2 Retention of Execution Traceability

The fourth requirement needs retention of the execution of traceability. We achieve this by modelling the cProv request and response as traceability graphs. Figure 5.8 shows the cProv request. It contains the agent (Bob) who is responsible for the making the request, the resource he/she is interesting in gaining access and the process involved.

```
RESPONSE-START
  entity(ex:e-responseId01), [cprovl:description=""])
  RESPONSE-DEC

    entity(ex:requestId01, [prov:type ='cprov:cResource']) //request Id
    agent(ex:policyLan01, prov:label="policy language") //policy Id

    entity(ex:resultId01, [prov:type ='cprov:cResource',
      cprovl:decision='cprovd:permit', cProvl:description=""])

    //processes for target and conditional statements
    activity(ex:target01,-, [prov:type='cprov:cProcess',
      cprov:userCloudRef="http://orangecloud/user@mufy/clusterX/imageX" %% xsd:anyURI,
      cprov:vProcessRef="http://platformX/ServiceX/ProcessX" %% xsd:anyURI,
      cprov:pProcessRef="http//ClusterX/ServerNameX/ServerIpX/ServerMacX/" %% xsd:anyURI])
    activity(ex:con01,-, [prov:type='cprov:cProcess',
      cprov:userCloudRef="http://orangecloud/user@mufy/clusterX/imageX" %% xsd:anyURI,
      cprov:vProcessRef="http://platformX/ServiceX/ProcessX" %% xsd:anyURI,
      cprov:pProcessRef="http//ClusterX/ServerNameX/ServerIpX/ServerMacX/" %% xsd:anyURI])

    entity(ex:resultId01, [prov:type ='cprov:cResource',
      cprovl:decision='cprovd:permit', cProvi:description=""])

    entity(ex:tr1, [prov:type='cprov:Transition', cprov:region='provd:GB',
      cprov:country='provd:England', cprov:latitude="51.5217 N",
      cprov:longitude="0.1012W", cprov:state='cprovd:origin', cprov:event='ex:ev001'])

    // relation edges
    wasDerivedFrom(ex:wdf01, ex:resultId01, ex:e-responseId01, -,-,-,
      [prov:type='cprov:wasReferenceOf', cprov:refType='cprovd:direct'])
    wasAttributedTo (ex:watId01, ex:resultId01, ex:policyLan01,
      [prov:type='cprov:hadOwnership', cprov:ownershipType='cprovd:possession'])
    wasInformedBy(ex:wib01 , ex:con01, ex:target01,
      [prov:type='cprov:wasExplicitlyCalledBy', cprov:explicitType='cprovd:validate',
      cprov:callComm='', cprov:callMedium='', cprov:callNetwork=''])
    wasGeneratedBy(ex:wvb01, ex:resultId01, ex:target01,-,
      [prov:type='cprov:wasVirtualizedBy',  cprov:purpose='cprovd:validated'])
    wasGeneratedBy(ex:wvb02, ex:resultId01, ex:con01,-,
      [prov:type='cprov:wasVirtualizedBy', cprov:purpose='cprovd:validated'])
    wasAssociatedWith(ex:wicb01, ex:target01, ex:policyLan01,
      [prov:type='cprov:wasInitiallyCalledBy', cprov:purpose='', cprov:accessMedium='',
           cprov:accessNetwork=''])
    used(ex:ex:used01, ex:target01, ex:requestId01, -)
    wasAttributedTo(ex:watId02, ex:tr1, ex:policyLan01,
      [prov:type='cprov:hadTransitionState, cprov:method='])

RESPONSE-END
```

FIGURE 5.7: Policy Response

Figure 5.9 shows the traceability graph for the policy response. It exhibits details of execution of policy response in relation to the policy request. A policyLanguage agent invokes the target process to match all the target statements, which in turn makes an explicit call to condition process to match the conditional statements.

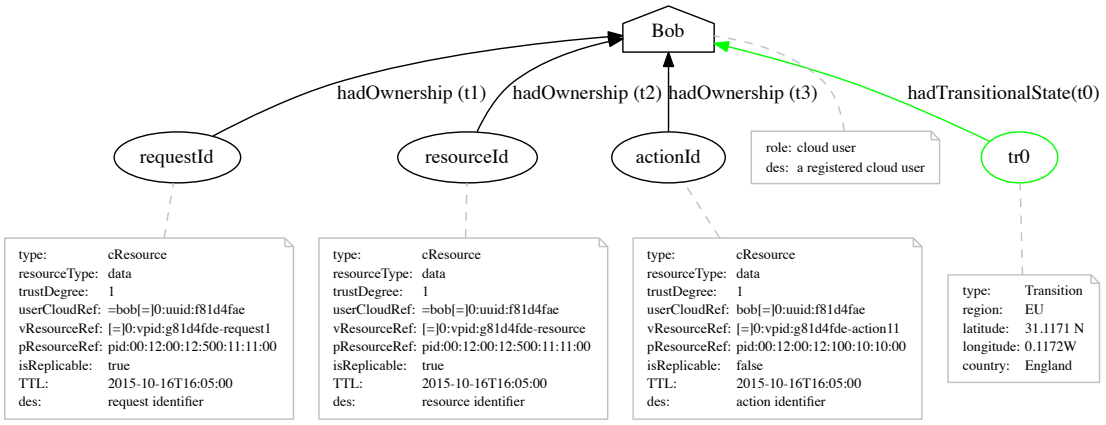


FIGURE 5.8: Policy Request Traceability Graph

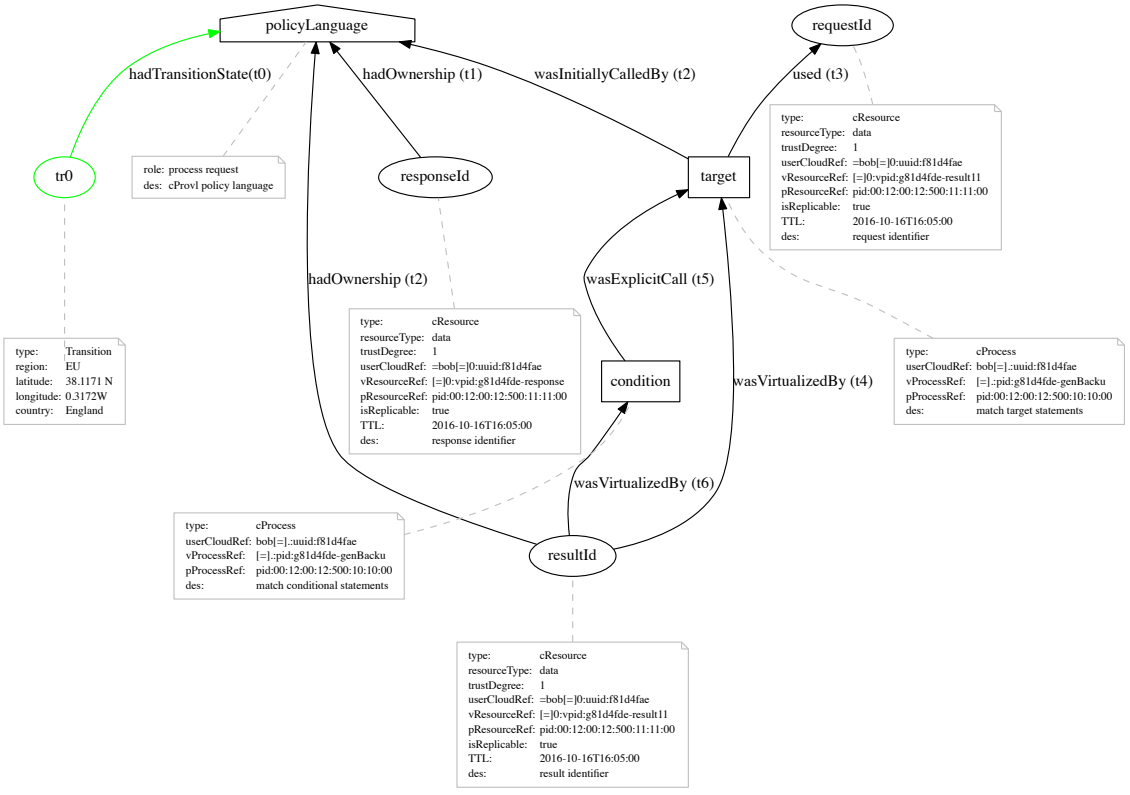


FIGURE 5.9: Policy Response Traceability Graph

5.4 Discussion

Our proposed policy language is capable of declaring policies and rules for capturing business requirements to enforce them. The language uses the Prov vocabulary for declaring statements, and defines its own grammar for the structure. The core aspects

of the language are the handling of traceability data for validating the policies, and use of variables to support dynamic value handling at run-time. This allows policies to be more generic and better adopted in a dynamic environment such as the cloud.

The syntax defined for presenting abstract policies is designed to provide a closer integration between the policy language and the traceability data modelling, which is also defined by the same notation. Having such a close integration enables us to create policies that can utilize full breadth of traceability graph. It would also be easier for users who are familiar with provenance to comprehend the language without having much difficulty. Moreover, we should be able to use existing Prov parsers (with some modifications to handle language specific syntax) to validate and verify policy for its correctness.

The cProvl policy language is inspired by the XACML standard and has similar structure but with additional support of traceability data. The vocabulary of the language is rich enough to capture relatively complex business rules.

The representation and selection of the provenance data plays an important role in determining the performance of the policy language. We therefore introduced the concept of different levels of granularity on the provenance data in the DEC section. This enables a user to select the right level of traceability data required for the validation, which by default is coarse grained (all traceability data selected). The effectiveness of the granularity is dependent on the representation of the traceability data.

The commercial adoption of such a language is highly dependent on two factors, compatibility and usability. A company should be able to integrate with the existing infrastructure without having to commit to major changes, and is able to model their requirements easily with minimal effort. The usability of the language can be simplified further by providing a high level UI wrapper that can reduce the effort of generating of the policy statements.

5.5 Summary

In this chapter a new provenance-based policy language (cProvl) has been proposed. The purpose of this language is to enable modelling of the service requirements, and be generic enough to be used by other services. The language introduces a number of innovative concepts, such as using the provenance data to validate the rules, use of variable for dynamic policy statement invocation at run-time, and retain provenance of the policy execution.

We have demonstrated a practical usage of the language by modelling one of the service level requirements (the remaining requirements are presented in Appendix A.7) and it should be flexible enough to model complex requirements of other services.

Chapter 6

Traceability Framework

So far in chapter four, we have defined a traceability model for the cloud which is based on W3C Prov. This work was extended in chapter five by proposing a provenance-based policy language that can express the service requirements as policies, and the language can validate its policies using traceability data. This chapter expands all this work by proposing a traceability framework which provides the necessary APIs for integrating the traceability model and policy language with a cloud based service. The aim of the framework is to help the developers in developing traceability-based enforcement cloud services. The framework handles the underlying complexity of capturing, storage of traceability and policy executions.

This chapter presents the architecture and design of the framework, which is designed to be scalable and is distributed in nature. It also provides a high level overview of the implementation of the framework, and its integration with the ConfidenShare service.

The contributions of this chapter are as follows:

- A traceability framework that provides client and server-side stacks for integrating service-level traceability and policy-based enforcement for existing and new cloud services. This enables seamless integration of all necessary components by using high-level APIs that hide the underlying mechanism for encoding and storing traceability data, invoking the enforcement based control of resource sharing, storage and processing of it.

- An implementation of the framework that leverages on industry standards (XML, XACML and Prov) for a greater degree of adaptability and compatibility with an existing service (ConfidenShare). The framework maps the cProvl policy language to the XACML standard that enables running of cProvl policies in an XACML compliance engine.

6.1 Motivation for a Traceability Framework

In the industry, there is an ever growing need for software to be produced and delivered in a short period of time with quality and desired features. To support such trends, many popular frameworks have appeared on the market, the likes of Struts [202], [117], Spring [19], and other MVC frameworks [180]. The commonality among these frameworks is they all mask the underlying technological complexities by providing a simplified high level interfaces that developers can interact with to develop and integrate their business logic. Based on personal experience, such advancement helps in minimizing the learning curve of using such frameworks, which can translate to increase in productivity and better industrial adoptions.

ConfidenShare service (Section 2.2) defines a number of business requirements, which require the support of traceability modelling and policy-based enforcement. An effective way of achieving this is by creating a framework that can adhere to the following requirements:

Requirement 6.1 : Ease of Integration

The integration of the framework should not require significant changes to the service infrastructure or alter the behaviour of the service. It should be as simple as using high level APIs or configuration files to define the integrations.

Requirement 6.2 : Leverage on the Industrial Standards & Solutions

It should leverage on the solutions that are currently deployed by businesses and are based on open standards. This would minimize the compatibility and adaptability stigma associated with new and emerging technologies.

Requirement 6.3 : Traceability Data Handling

The handling of traceability data should not add significant overhead to the running of a service, which needs to be generated and handled within a reasonable time.

Requirement 6.4 : Policy-based Business Rules Enforcement Handling

The framework should facilitate the business rules enforcement of a service on demand.

We now present the framework's architecture, design and its implementation.

6.2 Traceability Framework Stacks

It is imperative for the framework to provide ease of integration of the traceability model cProv and policy language (cProvl) with existing and new commercial-based cloud services. For this purpose, we have leveraged three standards: Prov and XML standardized at W3C and XACML at Oasis, which form the backbone of the framework's stacks. Figure 6.1 shows the client-side stack which is intended to interface with the server-side stack (see Figure 6.2).

6.2.1 Client Stack

The client stack handles operations such as the integration and generation of provenance data, as well as the request for provenance-based compliance. More concretely, it is structured as a five layered stack (see Figure 6.1).

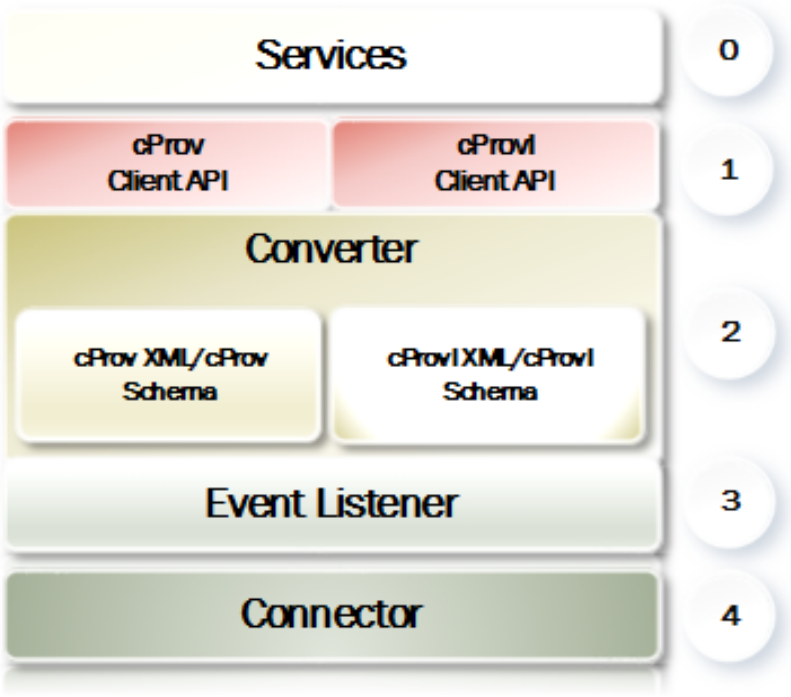


FIGURE 6.1: Framework Client Stack

Layer 0 - Defines the actual integration with a service. This is where one or more service clients are modified at the source code level to provide traceability capability. The service developers would explicitly incorporate the traceability and policy features to their selective service business logic (this has been applied to the ConfidenShare service (see Section 2.1)).

Layer 1 - Provides two APIs (traceability and policy) which define a list of capabilities

that are used by the service client to assist the generation of traceability data, and declaration of a request for policy compliance.

Layer 2 - Defines a list of converters (native to XML traceability and cProvl XML policy request) to encode the service-generated traceability statements into standardized XML traceability statements, as well as service-generated policy requests into XML-encoded policy requests. These are supported by the underlying schemas for cProv traceability model and cProvl policy request for their XML representations.

Layer 3 - Provides handlers to manage multiple generation of traceability statements and policy requests. It is intended to handle the traceability statements using a messaging queue (point-to-point) for them to be stored permanently in persistent storage remotely. Policy request is also handled in the same manner.

Layer 4 - Transfers the provenance statements to permanent storage and sends the policy request to the policy controller (on the server side) via HTTP.

6.2.2 Server Stack

The server side stack defines operations for storing, querying and updating the provenance store. For compliance control, it provides the mechanism for handling policy requests, translation and execution in the extended XACML policy engine.

The sever stack contains six layers as shown in Figure 6.2.

Layer 0 - Designed for creating server side modules for extending functionalities, such as a classifier, traceability graph visualizer or a statistical analysis module (currently no extended modules defined).

Layer 1 - Provides the server side integration of traceability and policy capabilities from the service client. It has two core APIs (cProv REST API and cProvl REST API). One for handling the provenance data and the other for compliance control.

Layer 2 -This layer supplies converters (cProvl to XACML, and XACML to cProvl) for interacting with the cProvl policy language engine. It handles conversion of cProv policy and request into XACML equivalent, and XACML response into cProvl response.

Layer 3 - Presents the cProvl policy engine that handles the cProvl policy requests. It extends the XACML engine to enable running of cProvl policies as XACML equivalent.

Layer 4 - Provides the mechanism for interfacing with the traceability and policy store.

Layer 5 - Defines the logical storage structure. It contains the traceability and policy store, that store traceability records and policies for services.

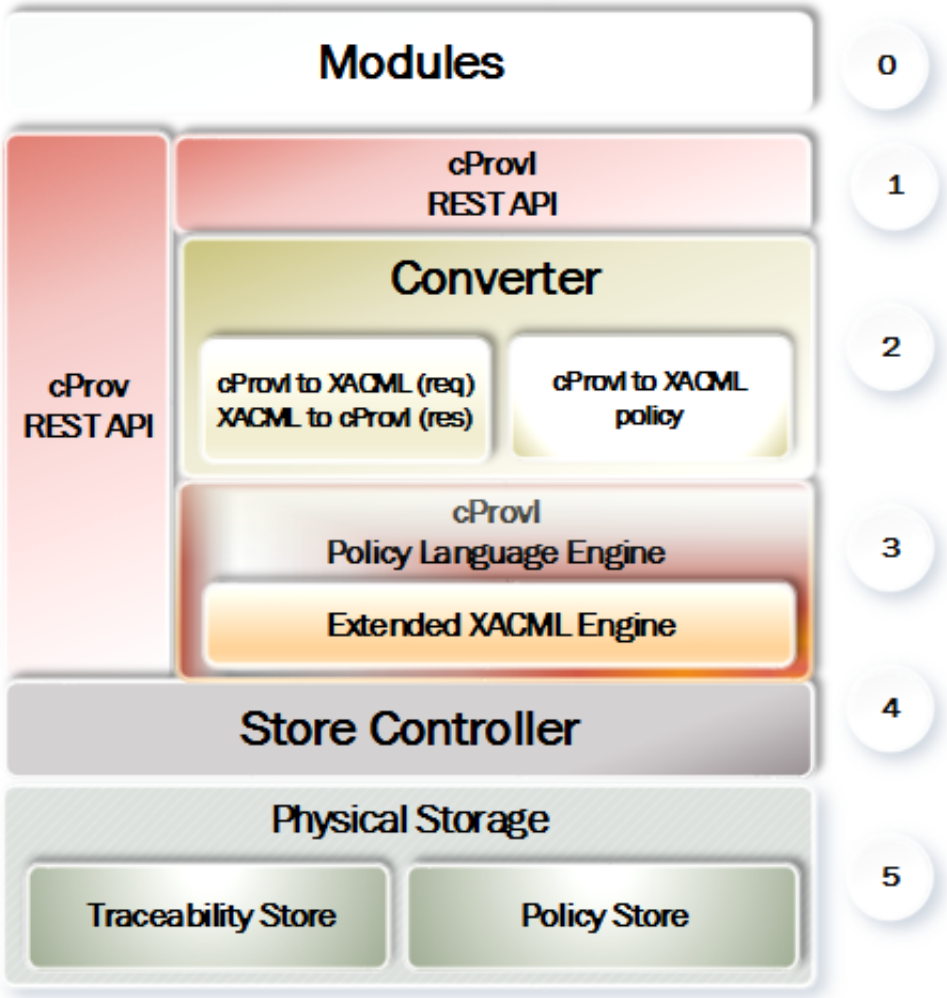


FIGURE 6.2: Framework Server Stack

By adopting these standards (Prov, XML and XACML), the framework is likely to be more compatible with the existing software development processes, tools and infrastructure.

XACML does not have any support for provenance, we address this deficiency by extending its core architecture to provide provenance support using our cProvl policy language. Section 6.2.3 below describes how we have extended its core architecture.

It is possible that having multiple layers (client and server stacks) may have an implication on the performance (see Section 7.2.2), however, this ensures that the stacks are low coupling, high cohesion and can scale. We believe this would aid toward ease of integration with a service.

6.2.3 Framework Architecture

From our literature review we can see the XACML architecture (see Figure 3.4) as being highly modular with extensibility in mind. It defines five core modules, which are designed to run independently in a distributed environment or stand alone. The modules are: PEP (Policy Enforcement Point), PDP (Policy Decision Point), context handler, PAP (Policy Administration Point) and PIP (Policy Information Point).

The starting point of the architecture is the PAP module, this is where policies are written and made available to the PDP module. A typical service would make its initial request to the PEP module. This request is then handled by the "Context Handler" in its native request format (XML). The "Context Handler" interfaces with the PDP, which is responsible for evaluating the request for making an appropriate decision. It provides all the necessary information required by PDP to make the correct decision. Once the decision is formulated and received (see Figure 3.4), it is sent to the PEP module. The PEP module evaluates the response, and optionally it interacts with the "Obligations Service" (which defines additional actions to be performed, see Figure 3.4) to fulfil the obligations [185].

The distributed nature of the XACML architecture, and the separation of concerns for each module makes it a good candidate for us to use as the base for our traceability framework architecture. Figure 6.3 shows the traceability framework architecture, which extends the five core components of the XACML architecture to support *traceability-based enforcement* of service business logic.

The PAP (writes XACML policies and makes them available to the PDP) module has been extended to allow the creation of cProvl policies, and provides a mapping from cProvl to XACML policies, as well as providing storage for them (1.a - 1d).

The PEP (handles the initial incoming service-specific request typically from an application) module has been extended to cater for a service request to be translated into cProvl request and stored in the policy store with its provenance (2-3). The service response is treated in the same manner (12.a-12.d).

The context handler is responsible for converting a service request into an XACML request. We provide the support for a cProvl request to be translated into an XACML request (4.a-4.d). The request is then transferred to the PDP module.

The PDP module determines the outcome of a request (11). We have introduced new functions to accommodate the handling of traceability data (used by the translated XACML polices). Before making a decision, it may request the context handler (5) for additional attributes via the PIP module (in our case, attribute references to traceability statements).

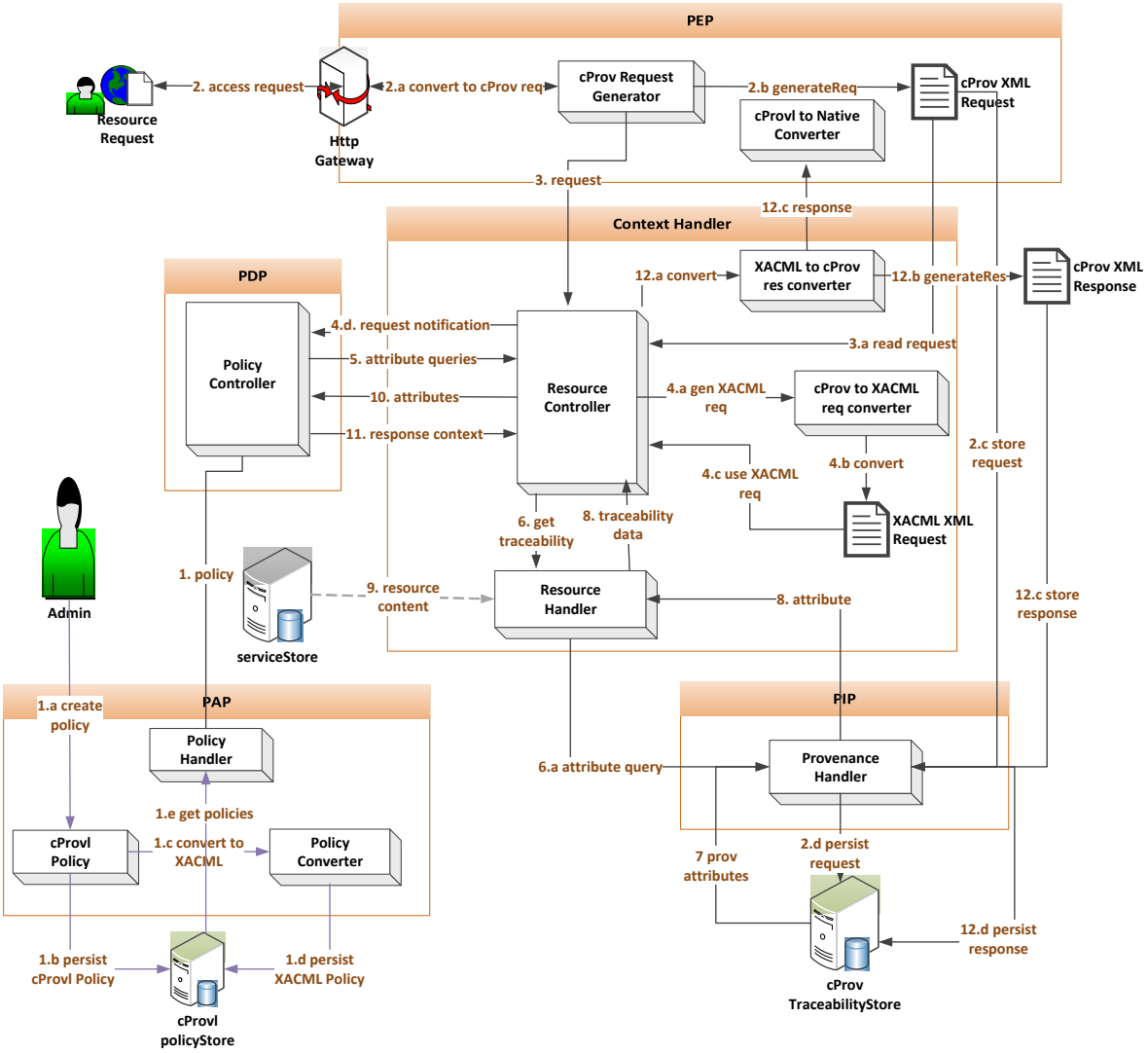


FIGURE 6.3: XACML Extended Architecture to Support Provenance

The PIP module has been extended to interface with the traceability store. It returns the necessary statements requested by the PDP module for decision making (6-8).

The context handler receives an XACML response from the PDP module. We have also added the support for an XACML response translated to a cProvl response (12.a-12.c) (stored in the policy store), which is then sent to the PEP module (12.c). The PEP translates it to service specific response and enforces the control, i.e. Permit/Deny.

6.3 Framework Design

In chapter four and five, we have used abstract syntax of prov-n for presenting the traceability model (cProv) and policy language (cProvl). For the implementation design, we have favoured XML/XML schema [33] as it is very mature and widely adopted in the industry. Given that XACML is also based on XML/XML schema, we have chosen to design the traceability model (cProv) and the policy language (cProvl) using XML/XML schema by extending the prov-xml schema [93] which is based on XML/XML Schema. This ensures that both (cProv and cProvl) can be easily extended, are not implementation dependent and can be integrated or mapped to other XML/XML schema based standards (XACML). The cProv schema inherits from the base level Prov elements and defines its own elements and attributes. The cProvl uses the cProv schema for declaring statements, and defines its own language syntax for creating provenance-based policies, rules, requests and responses.

The high level API (traceability store, policy-based enforcement) designs are based on the REST architectural style [72], which is increasingly becoming the de-facto standard for Web services. This provides loose coupling between the client and server, and can be scaled to handle increased load. On top of that, it can be easily integrated with multiple services without too much overhead.

We now present the cProvl traceability model encoding using XML schema.

6.3.1 cProv Traceability Model

As defined in Section 4.2, the cProv traceability model is based on the Prov traceability model, and uses its schema by extending it. The Prov schema acts as the base level elements for the cProv elements. Figure 6.4 shows the root element 'TraceabilityDocument', its sub-elements and inherited elements are from the Prov 'Document' element. Note, only selected elements structure are shown to illustrate how they extend the prov-xml schema and their composition. For full detailed schema elements, see Section A.1.

The root element extends the Prov 'Document' via inheritance; all the sub-elements of the 'Document' are accessible from the root element.

6.3.1.1 Entity Extension

The 'prov:entity' is the base element for the 'cprov:transition' and 'cprov:resource' elements. The 'cprov:transition' element is used to capture the state and location information for an entity, activity and agent during the execution of an event. The 'cprov:resource' is an abstract element.

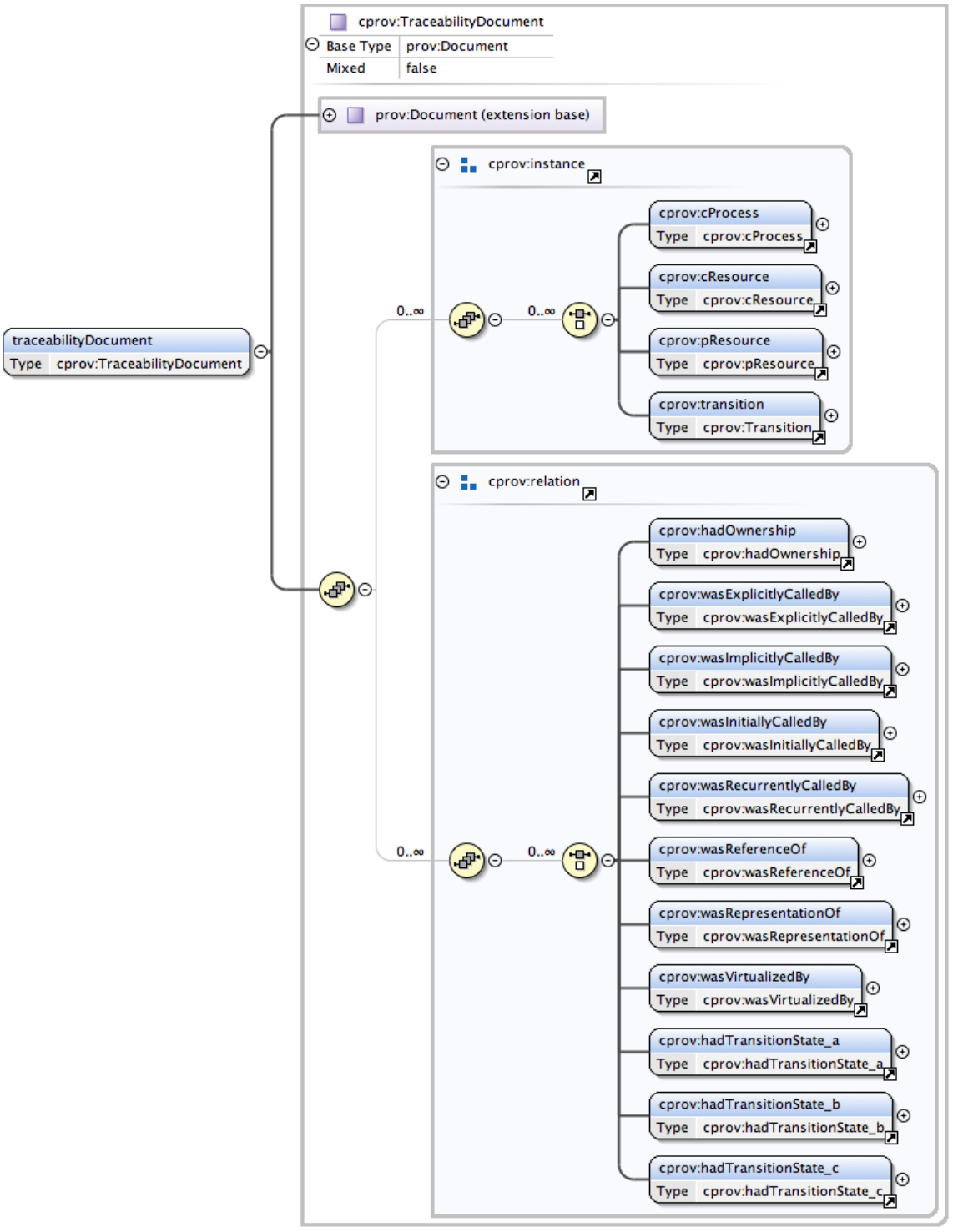


FIGURE 6.4: cProv- TraceabilityDocument Element

It is extended by the pResource and cResource. Figure 6.5 shows the cResource element's properties, such as 'cprov:userCloudRef', 'cprov:vResourceRef' and 'cprov:vResourceRef' which are mandatory. The 'cprov:TTL' and 'cprov:isReplicable' elements are optional.

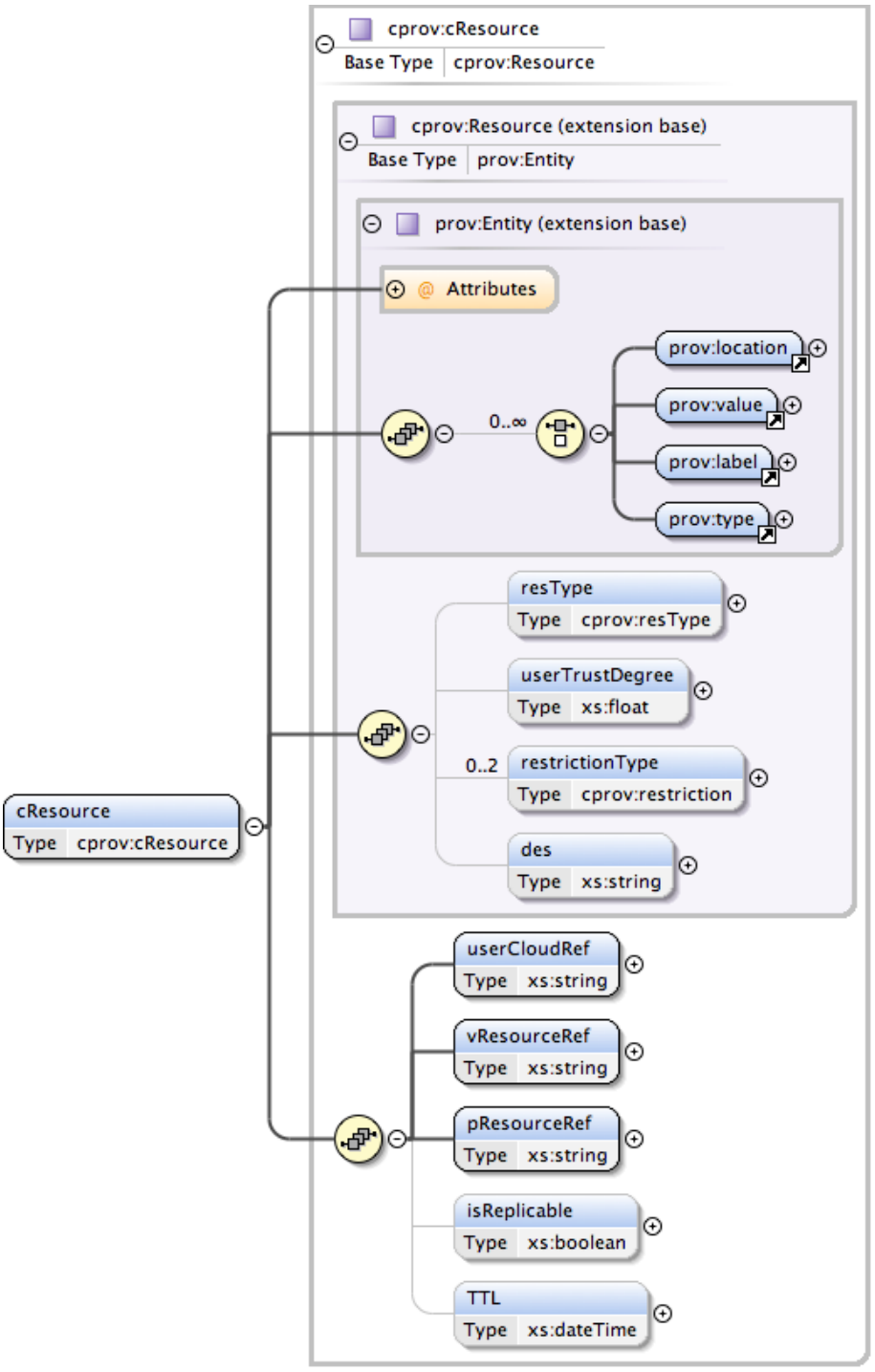


FIGURE 6.5: cResource Element

6.3.1.2 Activity Extension

The 'cprov:cProcess' element inherits from the Prov's 'prov:Activity' element, and defines three additional sub-elements, which are related to a cloud-based service (see Figure 6.6). They are 'cprov:userCloudRef', 'cprov:vProcessRef' and 'cprov:pProcessRef' used to capture the virtual-to-physical locality of a running process.

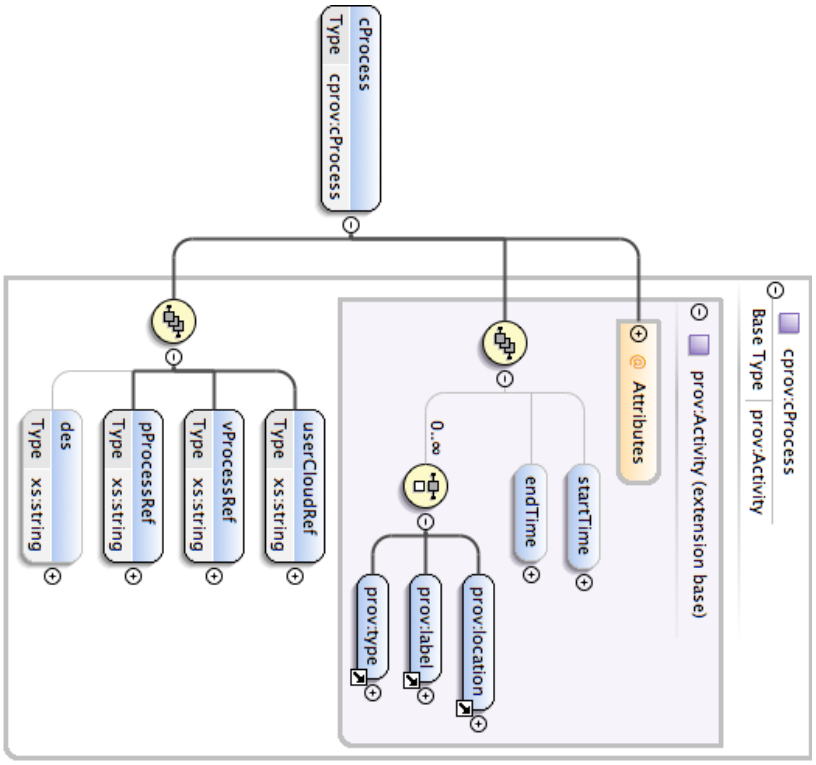


FIGURE 6.6: cProcess Element

6.3.1.3 wasInformedBy Extension

The base element 'prov:wasInformedBy' is sub-typed by the 'cprov:wasExplicitlyCalledBy', 'cprov:wasImplicitlyCalledBy' and 'cprov:wasRecurrentlyCalledBy'. Figure 6.7 shows the extended edge 'cprov:wasImplicitlyCalledBy' and its sub-elements, which are designed to express the characteristics of the invocation that took place between two processes for a cloud-based service.

6.3.1.4 wasAssociatedWith Extension

The 'prov:wasAssociatedWith' element is extended by the 'cprov:wasInitiallyCalledBy' and has additional sub-elements such as 'cprov:purpose', 'cprov:accessMedium' and 'cprov:accessNetwork'. They capture information related to the starting/running of a service. This can also be seen as the beginning or start of a running process or an application.

6.3.1.5 wasAttributedTo Extension

The base element 'prov:wasAttributedTo' is sub-typed by the element 'cprov:hasOwnership', which captures information related to the level of ownership of a resource.

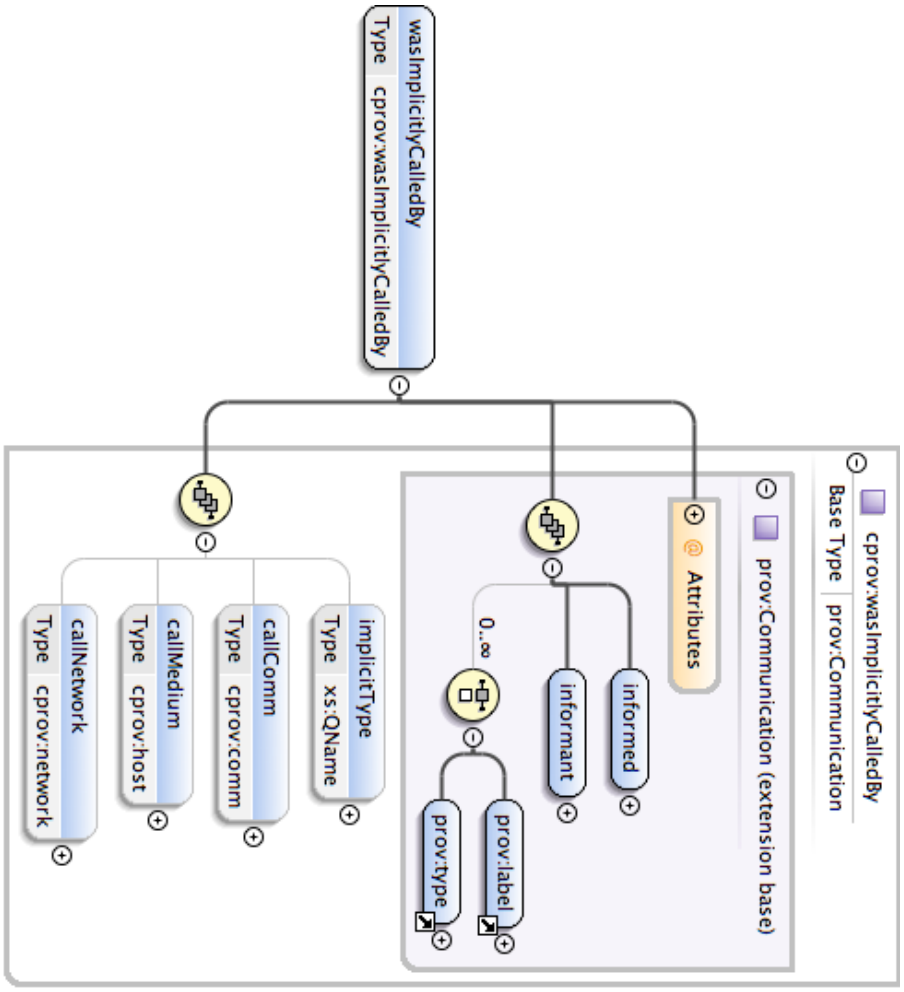


FIGURE 6.7: wasImplicitlyCalledBy Element

6.3.1.6 wasGeneratedBy Extension

The 'prov:wasGeneratedBy' base element is sub-typed by the elements: 'cprov:wasStatedBy' and 'cprov:wasVirtualizedBy', and they define their own specific elements (properties).

6.3.1.7 wasDerivedFrom Extension

The 'prov:wasDerivedFrom' base element is extended by the following elements: 'cprov:wasRepresentationOf', 'cprov:hadParticipatedAt' and 'cprov:wasReferenceOf'. They all indicate different variations of extensibility of an 'Entity', and captures its relevant properties.

6.3.2 Traceability Service Store API

The traceability store API (layer two in Figure 6.2) is designed to facilitate remote persistence operations for traceability data within the traceability store. It provides functionality such as create, store, update and query of provenance entries, which is quite similar to the public generic provenance repository ProvStore [95]. However, our store is more specific to XML-based traceability context of the ConfidenShare service and it is designed to be portable and run within the service environment. The API is based on the REST approach, which enables ease of integration with a service. We can define each of the operation more concretely, as follows:

The API currently has a one resource, *traceabilityDocument*, with a number of operations.

Resource URI

```
1 http://labs.orange.com/uk/cloudProvenance/cprov-ProvenanceStore/traceabilityDocument
2  /{serviceId}/{traceabilityType}/{documentId}/{elementId}
```

Resource Properties

Property	Description
ServiceId	Unique identifier of the service
traceabilityType	Type of traceability (Service, policy etc)
documentId	Unique identifier of the traceability document
elementId	Unique identifier of an element within the traceability document

A total of five operations defined for the resource.

6.3.2.1 Create Traceability Record

Operation Description - Creates a new traceability record. It is optional to add initial traceability entries, if none defined, a template traceability document is created and returns the Id of the new record.

Operation/Response

Resource Infomation	**Description**
Response Code	HTTP/1.1 201 Created
Response Error	HTTP/1.1 400 Bad Request
Response Body	documentId
Response Format	XML
Action	POST /traceabilityDocument/serviceId/traceabilityType

A successfully created resource would return a HTTP 201 code (corresponds to a created status), otherwise a HTTP 400 code, indicating a bad request.

An example:

```
REQUEST -  POST http://labs.orange.com/uk/cloudProvenance/cprov-ProvenanceStore/
      traceabilityDocument/ConfidenShare/ServiceTraceability HTTP/1.1
      Host: labs.orange.com
      Content-Type: application/xml
      Content-Length: nnn
      <cprov:TraceabilityDocument>  ... </cprov:TraceabilityDocument>

RESPONSE -  HTTP/1.1 201 Created

 <cprov:TraceabilityDocument>
   <cprov:cResource prov:id='ex:document001' cprov:des="http://labs.orange.com/uk/
         cloudProvenance/cprov-ProvenanceStore/traceabilityDocument/ConfidenShare/
         ServiceTraceability/document001">
    </cprov:cResource>
  </cprov:TraceabilityDocument>
```

This example shows the creation of a traceability record. The XML payload contains a traceability document which is appended to the resource body, and added to the created resource. The traceability record Id is returned (document001) and the URI to the resource.

6.3.2.2 Add Traceability Entry

Operation Description - To add traceability elements to an existing traceability record. The record is updated by appending the new elements at the end of existing elements.

Operation/Response

Resource Infomation	Description
Response Code	HTTP/1.1 200 OK
Response Error	HTTP/1.1 400 Bad Request
Action	PUT /traceabilityDocument/serviceId/traceabilityType/-documentId

A successful entry of a new element returns a HTTP 200 OK response, otherwise a HTTP 400 Bad Request is returned.

An example:

```
REQUEST -  PUT http://labs.orange.com/uk/cloudProvenance/cprov-ProvenanceStore/
      traceabilityDocument/ConfidenShare/ServiceTraceability/10062013 HTTP/1.1
      Host: labs.orange.com
      Content-Type: application/xml
      Content-Length: nnn

    <cprov:TraceabilityDocument>
          <cprov:cProcess prov:id='ex:a001'>
                ....
          </cprov:cProcess>
          <cprov:wasInitiallyCalledBy prov:id="ex:wicb0031">
                 <prov:activity prov:ref="ex:a001" />
                 <prov:agent prov:ref="ex:ag001" />
                ...
           </cprov:wasInitiallyCalledBy>
    </cprov:TraceabilityDocument>

RESPONSE - HTTP/1.1 200 OK
```

This example updates the traceability record called '10062013' by appending new elements (cProcess, wasInitiallyCalledBy) and the end of the collection. If successful, a response code 200 is returned.

6.3.2.3 Retrieve Traceability Record

Operation description - Retrieves a traceability record from the store. This would typically be an XML file containing all traceability elements for a specified record.

Operation/Response

Resource Infomation	**Description**
Response Code	HTTP/1.1 200 OK
Response Error	HTTP/1.1 404 Not Found
Response Body	Traceability Document
Response Format	XML
Action	GET /traceabilityDocument/serviceId/traceabilityType/-documentId

The HTTP GET method is used to retrieve the record. The desired record is defined as part of the REST URI structure. If the desired record is found, a HTTP 200 response and body containing the record is returned. Otherwise a 404 Not Found error code is return.

An example:

```
REQUEST - GET http://labs.orange.com/uk/cloudProvenance/cprov-ProvenanceStore/
      traceabilityDocument/ConfidenShare/ServiceTraceability/10062013 HTTP/1.1
      Host: labs.orange.com
      Content-Type: application/xml
      Content-Length: nnn

RESPONSE - HTTP/1.1 200 OK
    Content-Type: application/xml; charset=utf-8
    Content-Length: nnn

    <cprov:TraceabilityDocument>  <!-- multiple traceability elements -->
    </cprov:TraceabilityDocument>
```

This example requests for an entry called '1006213' to be retrieved. The response content is an XML document containing all the traceability elements for that record.

6.3.2.4 Retrieve Current Traceability Record ID

Operation description - Get the current or most recent traceability record Id.

Operation/Response

Resource Infomation	Description
Response Code	HTTP/1.1 200 OK
Response Error	HTTP/1.1 404 Not Found
Response Body	Traceability Document
Response Format	XML
Action	GET /traceabilityDocument/serviceId/traceabilityType/-documentId/elementId

The HTTP GET method is used to retrieve the traceability record Id. It will try to retrieve the currently created or used traceability record Id. If the non found, a 404 Not Found error code is return.

An example:

```
REQUEST - GET http://labs.orange.com/uk/cloudProvenance/cprov-ProvenanceStore/
      traceabilityDocument/ConfidenShare/ServiceTraceability HTTP/1.1
      Host: labs.orange.com
      Content-Type: application/xml
      Content-Length: nnn

RESPONSE - HTTP/1.1 200 OK
    Content-Type: application/xml; charset=utf-8
    Content-Length: nnn

    <cprov:TraceabilityDocument>
          <cprov:cResource prov:id='ex:document001'> ...</cprov:cResource>
    </cprov:TraceabilityDocument>
```

This example requests for ID of the current of most recent traceability record used to be returned. The response content is an XML document containing the ID (ex:document001) for that record.

6.3.2.5 Retrieve Traceability Element

Operation description - Retrieves a traceability element based on the element Id. A successful match returns a record with the matched entry.

Operation/Response

Resource Infomation	**Description**
Response Code	HTTP/1.1 200 OK
Response Error	HTTP/1.1 404 Not Found
Response Body	Traceability element
Response Format	XML
HTTP Type	GET /traceabilityDocument/serviceId/traceabilityType/-documentId/elementId

If there is an element that matches the Id in the traceability record, it is returned in the body of response with a HTTP 200 OK code. Otherwise a 404 Not Found element is returned. An example:

```
REQUEST - GET http://labs.orange.com/uk/cloudProvenance/cprov-ProvenanceStore/
      traceabilityDocument/ConfidenShare/ServiceTraceability/10062013/ex:e001 HTTP/1.1
      Host: labs.orange.com
      Content-Type: application/xml
      Content-Length: nnn

 RESPONSE -  HTTP/1.1 200 OK
    Content-Type: application/xml; charset=utf-8
    Content-Length: nnn

    <cprov:TraceabilityDocument>
          <Entity prov:id='ex:e001'>  ... </Entity>
    </cprov:TraceabilityDocument>
```

This example searches for an element 'ex:e001' within the '1006213' record. Note, the URI must be encoded to handle the namespace. If successful, the matched element is returned with a HTTP 200 response code.

6.3.3 Policy-based Enforcement

Policy-based enforcement is designed for an application to adhere to certain business logic constraints. These constraints are defined as policies and enforcement is applied based on validating against traceability data. Figure 6.8 shows the overall design of the policy based enforcement for a service. It consists of three layers: Application, Policy and Persistent. Application references to layer 1 of the server stack (see Figure 6.2), Policy to layers 2-3 and Persistence to layers 3-4.

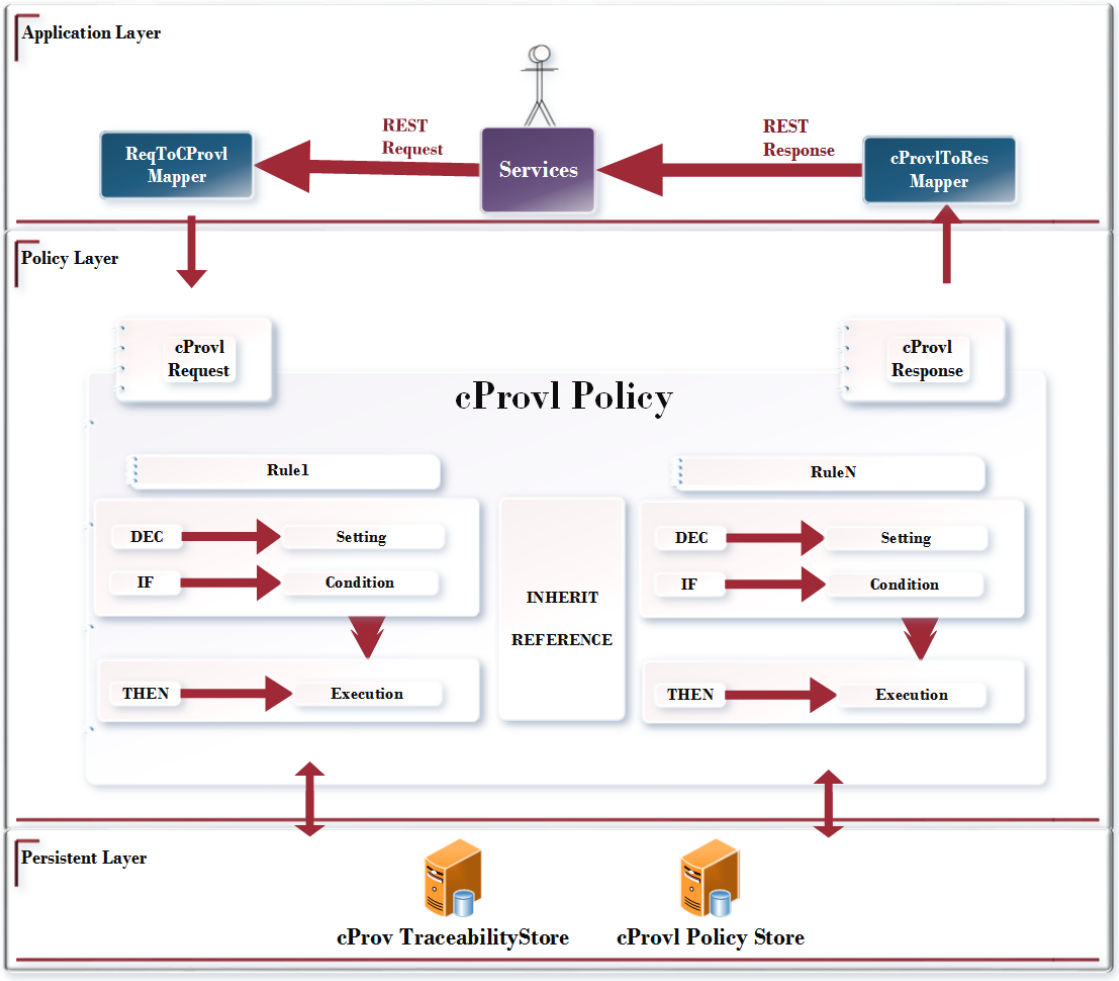


FIGURE 6.8: Policy-based Enforcement Design

Application Layer - The application layer is responsible for handling the request and response of a service. A service user makes a request (e.g. share resource via REST web service), which is translated into cProvl request. The encoding request is then sent to policy layer.

Policy Engine Layer - The policy layer selects the necessary policy and validates the request via the persistence layer, and produces a response with is sent back to the application layer (see Figure 5.1).

Persistent Layer - The persistent layer is responsible for querying and storing of traceability data and policies.

6.3.4 Policy-based Enforcement Language (cProvl)

The cProvl policy schema defines a structure for creating policies that capture the service level business logic. The policies enable enforcement of control based on the permitted operations. It leverages on the cProv traceability model and Prov schema, and provides

its own vocabulary elements with necessary fields.

6.3.4.1 Policy Element

Figure 6.9 shows a graphical representation of the XML schema element 'cprovl:policy' for the cProvl policy language. It consists of two elements which need to occur in a particular order: entity (that entails the unique identification of the policy and a brief description) followed by one or more rules (denoted by 1..∞).

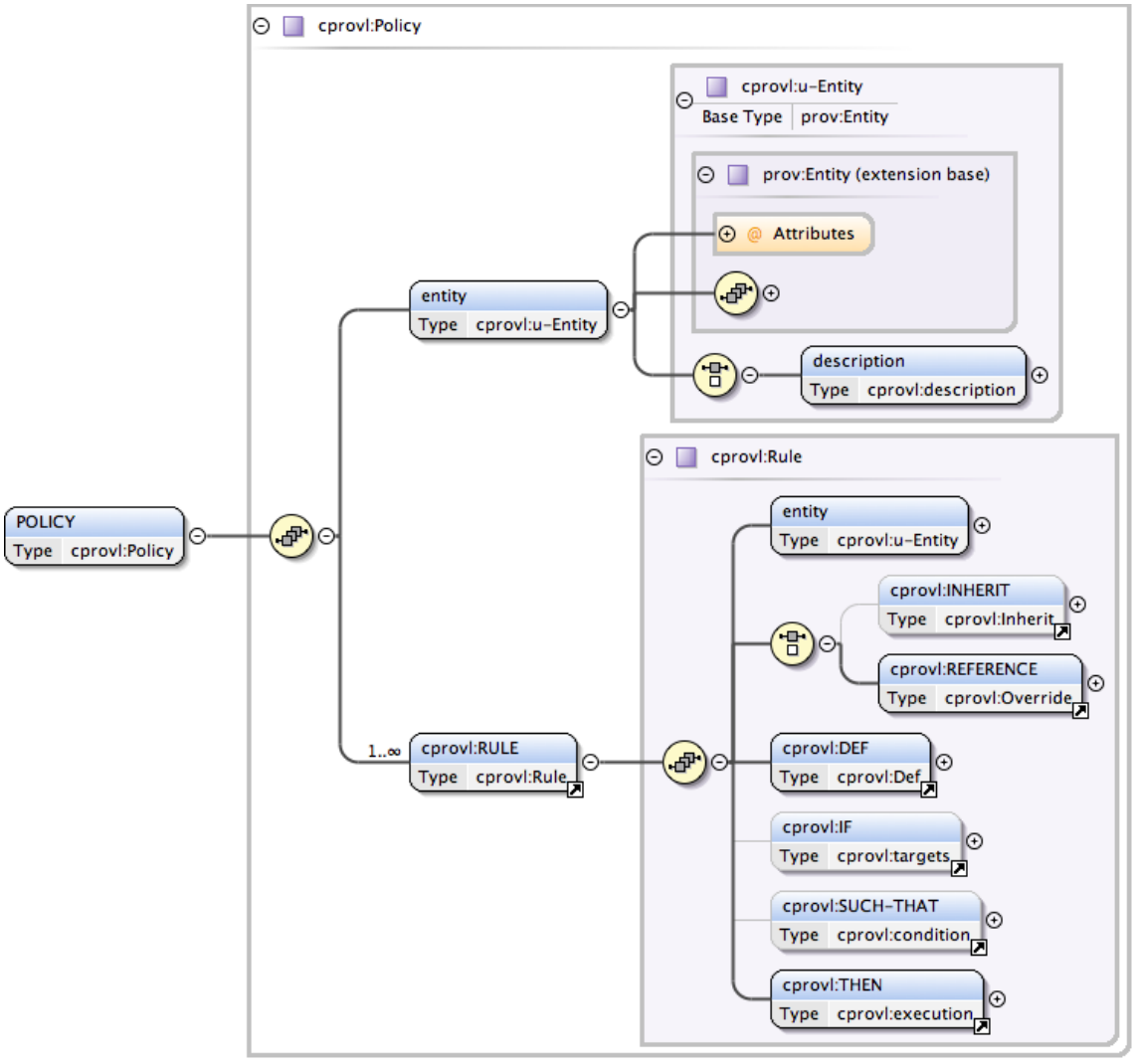


FIGURE 6.9: cProvl- Schema Element Policy

Each policy is created as an individual XML file. Multiple policies are not permitted in a single XML policy file. However, it is valid for multiple rules within a policy.

6.3.4.2 Rule Element

A rule element consists of a unique identifier defined by an entity element. The three key sub-elements of the rule are the cprovl:IF, cprovl:SUCH-THAT and cprovl:THEN.

IF element

The IF element (see Figure 6.10) denotes the start of a rule expression (target statements), and has an attribute 'quantifier'. It takes only two values 'for all' and 'there exist'. It is followed by optional var element, and IDs.

The 'var' element contains two attributes 'cprovl:var_type' and 'cprovl:var_identifier'. The 'type' can be either 'new' or 'ref'. An 'identifier' is of type xsd:Qname, and can take a reference to any of the cProv/Prov nodes/edges of the ID.

It also allows boolean operators, where more than one IDs or variables are required. The boolean operator is needed to be placed in between the multiple IDs.

More expressive rules are defined using the conditional statements in the 'SUCH-THAT' clause.

SUCH-THAT element

The SUCH-THAT element (see Figure 6.11) defines the additional conditions of a rule. These conditions are defined using statements. A statement can be of any type of cProv/Prov nodes or edges. It is possible to negate a statement using the attribute 'operator'. If there are more than one statements, a boolean or conditional operator can be placed in between them. One or more statements can be grouped using the Grouping element.

THEN element

The THEN element (see Figure 6.12) defines the outcome of the policy execution.

The outcome is defined as an entity, that contains action and resource elements. Action element provides the type of action to be taken (permit, deny, indeterminate and non-applicable). The obligation elements are used to define type of obligations to be taken. All entities are associated with the 'hadOwnership' element.

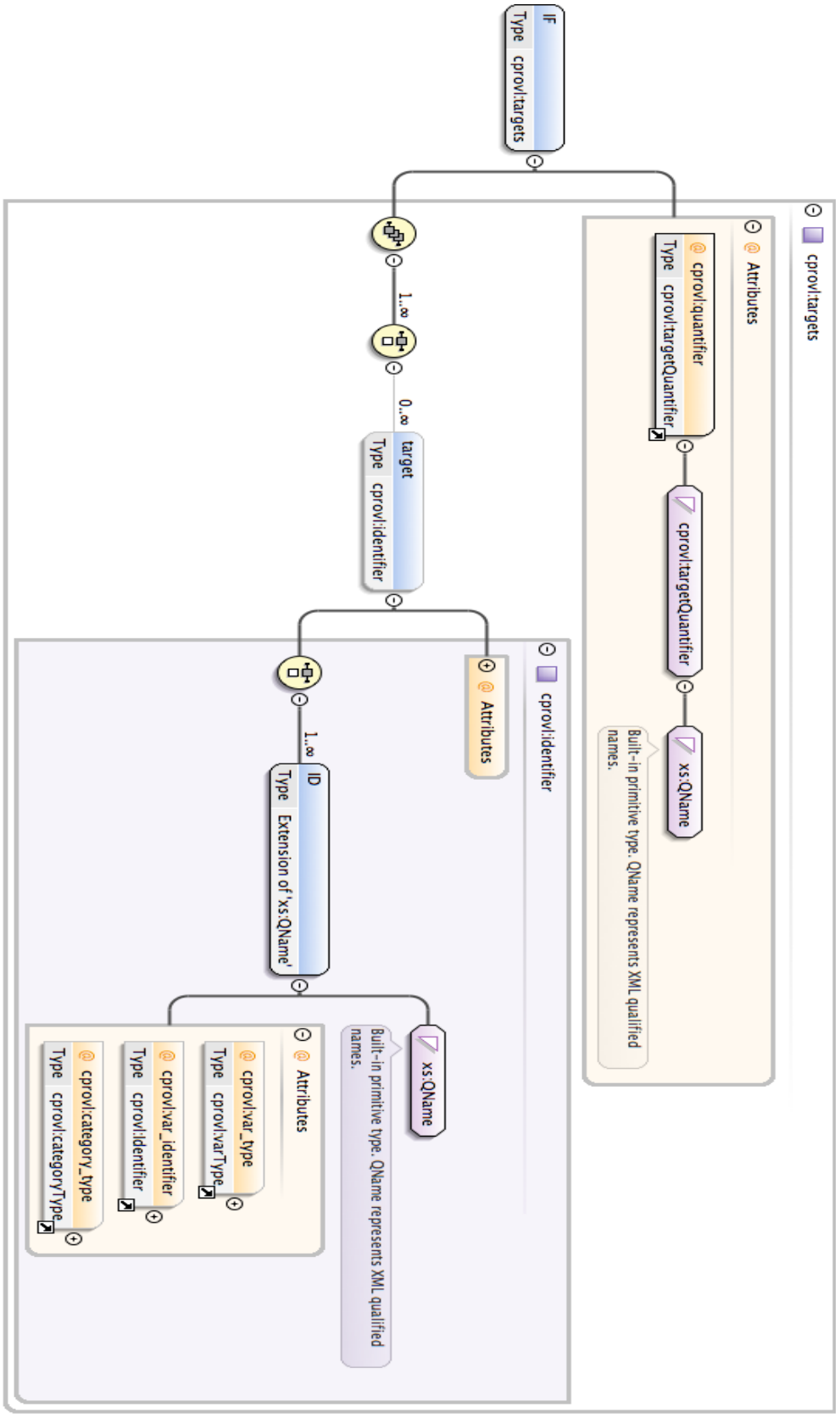


FIGURE 6.10: cProvl-Schema Element IF

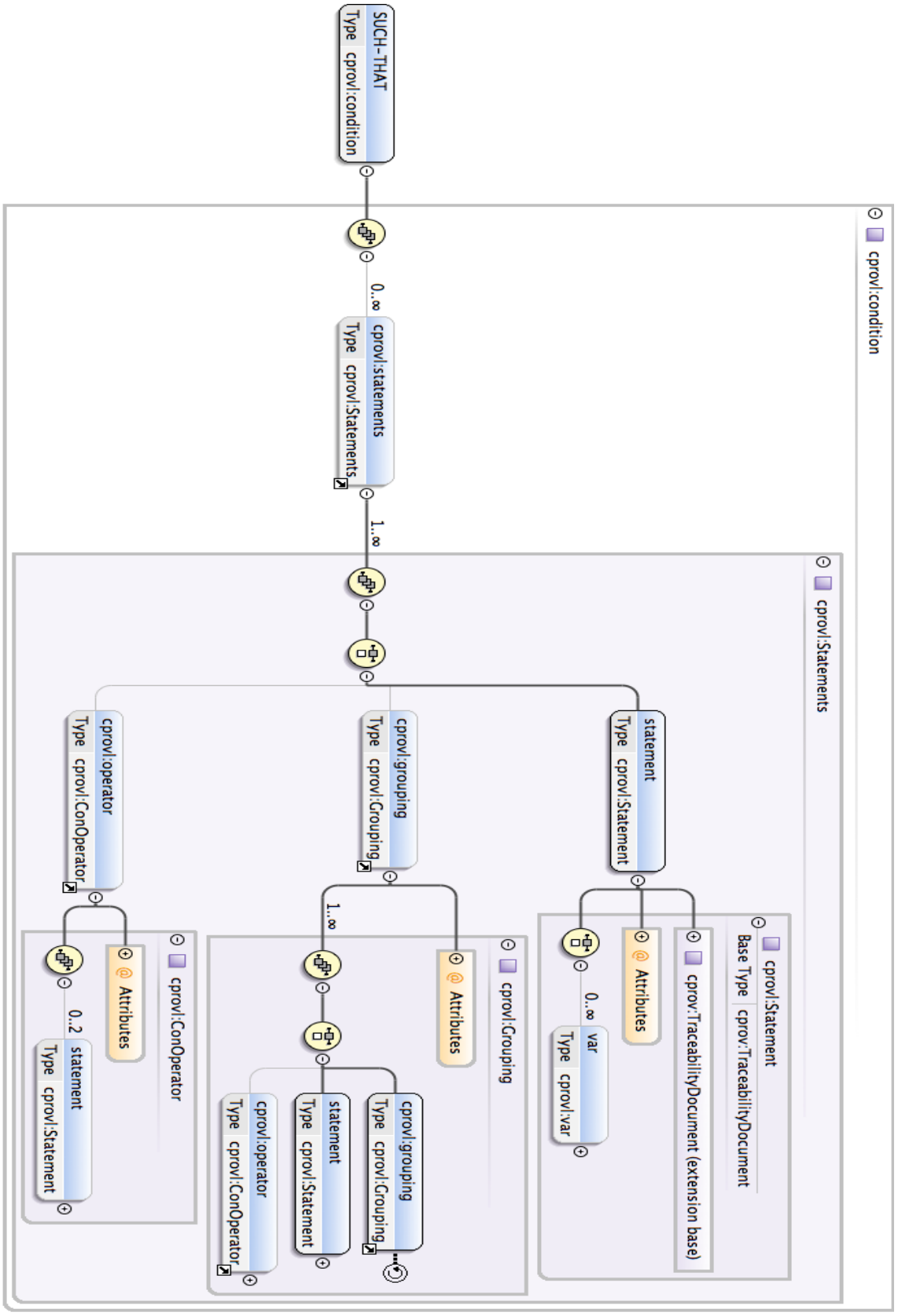


FIGURE 6.11: cProvl-Schema Element SUCH-THAT

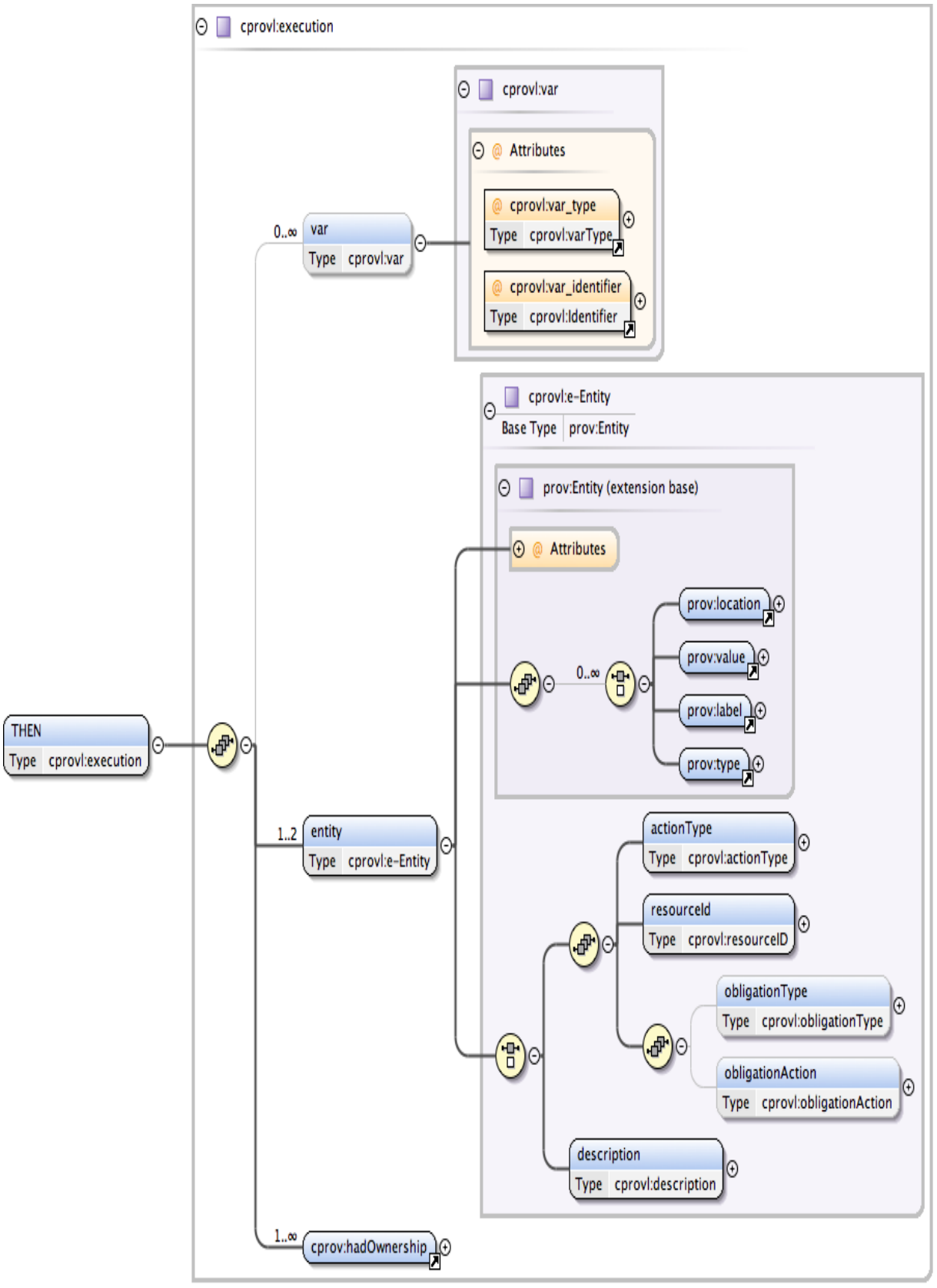


FIGURE 6.12: cProvl-Schema Element THEN

6.3.5 Policy Enforcement API

The policy enforcement API (references layer 1 on the server side stack, see Figure 6.2) allows a service to make a provenance-based policy request (such as a request to access or sharing a resource). The API handles all the underlying complexity of encoding and sending the request to the server end, and returns a response that the service acts upon.

The Policy Enforcement API is based on REST approach. It has two URI schemas, one for posting a request to the server, and another for getting the response back from the server.

Resource URI

```
/****** posting a policy request *********/
http://labs.orange.com/uk/cloudProvenance/PolicyController/policyEnforcement/{serviceId}

/******* getting a server response **********/
http://labs.orange.com/uk/cloudProvenance/PolicyController/policyEnforcement/{serviceId}/
/{responseId}
```

Resource Properties

Property	Description
ServiceId	Unique identifier of the service making the request
responsetId	The identifier used to retrieve the response for the policy request

6.3.5.1 Service Policy Request

Operation Description - A service user can make requests to gain access to one or more resources, or to attain a certain level of privileges.

Operation/Response

Resource Infomation	Description
Response Code	HTTP/1.1 200 OK
Response Error	HTTP/1.1 400 Bad Request
Response Body	requestId and responseId
Response Format	XML
Action	POST /policyEnforcement/serviceId

An example:

```
REQUEST- POST http://labs.orange.com/uk/cloudProvenance/PolicyController/
      policyEnforcement/ConfidenShare HTTP/1.1
      Host: labs.orange.com
      Content-Type: application/xml
      Content-Length: nnn

 <cProvl:policyRequest>  ..... </cProvl:policyRequest>

RESPONSE -  HTTP/1.1 200 OK
    Content-Type: application/xml; charset=utf-8
    Content-Length: nnn

<cprov:TraceabilityDocument>
 <cprov:cResource prov:id='ex:req001'> </cprov:cResource> <!-- request Id -->
   <cprov:cResource prov:id='ex:res001' cprov:des="http://labs.orange.com/uk/
       cloudProvenance/PolicyController/policyEnforcement/ConfidenShare/res001">
   <!-- responseId with response URI -->
   </cprov:cResource>
 </cprov:TraceabilityDocument>
```

This example shows a cProvl policy request payload is posted to the server, and a requestId and responseId are returned, the responseId is used to retrieve the policy response.

6.3.5.2 Service Policy Response

Operation Description - The response is designed to retrieve the outcome to the cprovl policy request from the server.

Operation/Response

Resource Infomation	**Description**
Response Code	HTTP/1.1 200 OK
Response Error	HTTP/1.1 400 Bad Request
Response Body	cProvl policy response
Response Format	XML
Action	GET /policyEnforcement/serviceId/responseId

An example:

```
REQUEST- GET http://labs.orange.com/uk/cloudProvenance/PolicyController/
      policyEnforcement/ConfidenShare/res001/ HTTP/1.1
      Host: labs.orange.com
      Content-Type: application/xml
      Content-Length: nnn

RESPONSE -  HTTP/1.1 200 OK
    Content-Type: application/xml; charset=utf-8
    Content-Length: nnn

 <cProvl:policyResponse> ......    </cProvl:policyResponse>
```

This example shown the responseId (res001) is used to get a response from the server. Based on the response, the service would either permit or deny the request from the user. The response is in essence a traceability graph (an example, see Figure 5.9), which is stored in conjunction with the request in the cProvl Traceability Store (see Section 6.4.3 for more details).

6.4 Framework Implementation

The right choice of implementation technologies are paramount for the adoption of the framework by the industry. The use of industrial standards and mature technologies are necessary for providing ease of integration with the existing service infrastructure, and for minimizing the learning curve of using such framework.

The framework implementation language of choice is Java [83]. It is a well established, mature and relatively secure language extensively used in the industry. It has a large range of open-sourced projects, technologies and frameworks available with large community support. Java related technologies such as JAXB [75], JAXP [137], JAX-RS [115], Spring [100] and ActiveMQ (JMS) [91] are used to develop the backbone of the framework, that includes processing and handling of XML [36], REST [171] based APIs and interactions amongst the framework components. The back-end server is based on the largely adopted Tomcat Server [120].

The core part of the framework is the cProvl policy language. Our approach to implementing the policy language has been to leverage on the latest industrial standard XACML 3.0 [185]. The open source implementation of this standard is the Balana [48] by WS02. XACML 2.0 [150] is currently the widely supported standard in the industry. However, our belief is that XACML 3.0 is a much richer standard than XACML2 and will supersede the previous standard within the next few years.

6.4.1 Traceability Support for XACML 3.0

We have added the traceability support to the XACML 3.0 Balana engine by extending some of its core functionalities. Figure 6.13 shows the extended functionalities required for the XACML engine to execute a cProvl-based policy and request. The engine adds an additional layer ('Translator') for mapping and generating an XACML policy and request from a cProvl policy and request.

To support the core traceability data for decision making, additional functions are required to address the following challenges:

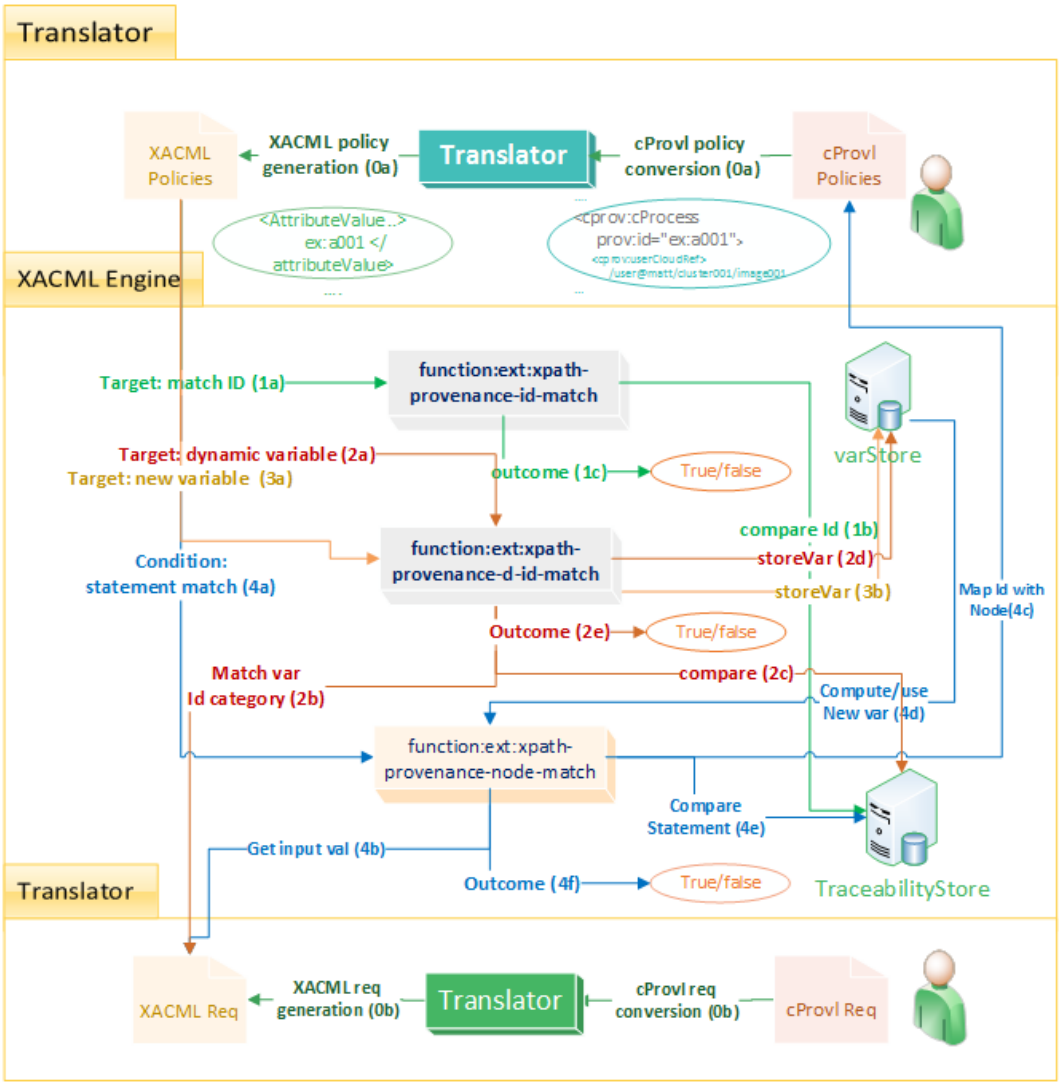


FIGURE 6.13: XACML Policy Engine Extension

6.4.1.1 Coupling of Policy Assertions

XACML policies are by default tightly coupled with requests; the policy's target and condition statements are primarily dependent on the values obtained from the request

to grant or deny access. cProvl, on the other hand, is more loosely coupled, as it is not solely dependent on the request values. While it may take some values from the request, policies are primarily focused on the data from the traceability store for its assertions. This requires XACML XPath functions to operate on the traceability store. However, they are restricted to 'content' XML from the request. To overcome this issue, we have introduced a new function for target:

labs:orange:com:function:ext:xpath-provenance-id-match

The target statements (defined as IDs, see Section 5.2.3.3) are handled by this function (it matches against the traceability store (see 1a, 1b and 1c on the diagram 6.13).

6.4.1.2 Dynamic Variable Holder

cProvl require dynamic variable holders for its statements. One statement may generate reference IDs stored in a variable, which is later required/used by another statement. *Such a concept is not present in XACML.* In order to address this issue, we have introduced another function:

labs:orange:com:function:ext:xpath-provenance-d-id-match.

The 'd-ref', 's-ref' and 'new' variables are declared in the target section of the policy. The d-ref values are extracted from the input request type (e.g. subject), matched against the traceability store and stored in the varStore (temp storage area for variable values). The s-ref values on the other hand are extracted from the input request and stored in the variable store (see 2a-2e).

For 'new' variable, it creates an entry in the store for the statement assertion (see 3a-3b). The content are of a new variable populated and used by the conditional statements.

6.4.1.3 Single to Multi-value Mapping

XACML conditional statements are single value entry attributes, whereas cProvl statements are multi-valued nodes/edges. In order to map single-to-multi-values, we have created a third new function:

labs:orange:com:function:ext:xpath-provenance-node-match.

This function first obtains the attribute value of an XACML policy conditional statement (this value needs to be a unique ID). It uses this as an xPath reference to a node in

cProvl Policy. If a match is found, the node and its properties are matched against the traceability store. If all is successful, it returns true, otherwise false (see 4a-4f). A variant of the function called labs:orange:com:function:ext:xpath-provenance-nodes-match is also created to handle equality of two nodes of same type. If they are same (their attributes value) it returns true, otherwise false.

6.4.2 XACML Policy Generation

The XACML policies are generated dynamically by the 'cProv to XACML' converter. It uses the cProvl policy as the input; an XSLT stylesheet transformer converts it to an equivalent of XACML policy.

The conversion process requires mapping of the rules, targets, conditions, logical operators and execution of cProvl policy into an XACML equivalent. Where direct mapping is not available, equivalent mapping is provided. Please see below (Figure 6.14) a simplified view of the stylesheet that is developed for carrying out such policy conversions. There are also stylesheets used for translating a cProvl request into an XACML request, and XACML response to a cProv response (see Section A).

For presentation purposes, we have split the policy stylesheet into two parts: first showing the translation up to the target statements using a cProvl policy which is converted into XACML equivalent. The second part shows the stylesheet from the end of target statements (beginning of conditional statements) to the end of the policy.

XSLT Stylesheet - Up to the Target Statements

The XSLT Stylesheet below (Figure 6.14) transforms a cProvl policy up to the target statements and it is used as the input by the stylesheet to produce an XACML equivalent as output. Note, the stylesheet has been edited for readability purposes (full stylesheet can be seen in Section A.3).

Lines 4-14 convert the policy and rule details (such as id, description, and rule effect (permit/deny)) into XACML comparable statements. Lines 15-26 handle the mapping of cProvl IDs into equivalent XACML targets.

cProvl & XACML Policy - up to the Target Statements

Figure 6.15 shows a cProvl policy (shows upto the target statements cprov:IF) used as the input to generate XACML policy (up to targets).

The cProvl policy (lines 2-15) defines two variables and an ID as target statements. The first is a resource variable (r:e001), its value is read from the input request (denoted by the cprovd:d-ref) (lines 7-9) and stored in the variable. The values from the variables are used in the condition statements (see below Figure 6.17). The second variable is a subject variable (r:ag001), its value is also read from the input request (lines 10-12)

```
1  <xsl:stylesheet xmlns="..xacml:3.0.."..>
2   <xsl:output method="xml" ../>
3   <xsl:template match="/">
4     <Policy xmlns="..xacml:3.0.."..> ..
5     <Target/>
6      <xsl:for-each select="cprovl:POLICY/cprovl:RULE">
7       <Rule RuleId="Rule1" Effect="Permit"> ..
8        <xsl:attribute name="Effect">
9         <xsl:variable name="actionEntry" select= "cprovl:THEN/
10            cprovl:entity/cprovl:actionType"/>
11        <xsl:value-of select="tokenize($actionEntry, ':')[position()=2] "/>
12          </xsl:attribute>  ...
13        <xsl:if test="count(cprovl:IF)!=0"> <Target>
14        <!-- Find all IDs, and variables -->
15        <xsl:for-each select="cprovl:IF/cprovl:target">
16         <xsl:variable name="fieldType" select="child::cprovl:ID/@cprovl:category_type"/>
17         <xsl:variable name="varType" select="child::cprovl:ID/@cprovl:var_type"/>
18         <xsl:if test="$varType='cprovd:new' ">
19            <AnyOf> <AllOf>
20             <Match
21              MatchId="..xpath-provenance-d-id-match">
22              ... <!-- set data type, Category, value -->
23            </Match></AllOf></AnyOf></xsl:if>
24        </xsl:for-each> </Target>
25         ...
26      </xsl:if>
```

FIGURE 6.14: cProvl Policy to XACML Policy Stylesheet - up to Target Statements

and stored in the variable. The third is an action ID. Its value is validated against the traceability store (denoted by the variable type cprovd:d-ref) (lines 13-15).

Three target statements are generated in the XACML policy (lines 19-46). The first (lines 27-37) target is equivalent to the lines 7-9 of the cProvl policy. They use a custom function (xpath-provenance-d-id-match, see Section 6.4.1.2) to handle the variable functionality of the cProvl policy language. This function maps the resource value as an xPath from the policy request, and stores it temporarily in memory to be used by the conditional statements. Lines 38-48 is the equivalent of 10-12 of the cPolicy and uses the same function. Lines 22-27 is the result of mapping of (lines 13-15) of cProvl to XACML. It uses the function (xpath-provenance-id-match, see Section 6.4.1.1) to validate the value against the traceability store.

We now present the second part of the stylesheet that translates from the condition statements to the end of the policy.

XSLT Stylesheet - From the Conditional Statements to End of the Policy

Figure 6.16 shows the XSLT stylesheet that provides mapping from the conditional statements to end of a cProvl policy to an XACML policy.

The stylesheet iterates through each of the conditional statements and checks for the boolean and conditional operators and groupings. It then generates the necessary equivalent XACML and custom functions.

cProvl & XACML Policy - From Conditional Statements to End of the Policy

From the Figure 6.17, the cProvl policy (lines 61-105) defines three conditional statements. The first is the cprov:hadOwership edge with its associated properties. As it can be seen from line 64, there is an entity that references and uses the variable (r:e001)

```
1  <!-- ****************  cProvl policy - up to target statements *************************-->
2  <cprovl:POLICY xmlns:cprov="http://orangelabs.com/cprov#" .... cProvl/cProvl-v1.0.xsd">
3   <cprovl:entity prov:id="ex:policy1"> ... </cprovl:entity>
4    <cprovl:RULE>
5     <cprovl:entity prov:id="ex:rule1"> <cprovl:description>Re-sharing of data is not permitted </cprovl:description>
6     </cprovl:entity> ....
7     <cprovl:IF cprovl:quantifier="cprovd:There-exist"> <!-- Check for any input request and match it with the store -->
8      <cprovl:target>
9       <cprovl:ID  cprovl:var_type="cprovd:d-ref" cprovl:var_identifier="r:e001" cprovl:req_field="cprovd:Resource">r:e001</cprovl :ID>
10     </cprovl:target> <!-- Check for the request user -->
11     <cprovl:target>
12      <cprovl:ID cprovl:var_type="cprovd:d-ref" cprovl:var_identifier="r:ag001" cprovl:req_field="cprovd:Subject">r:ag001</cprovl:ID>
13     </cprovl:target><!--  Process for sharing  -->
14     <cprovl:target>
15      <cprovl:ID  cprovl:req_field="cprovd:Action">ex:a-share</cprovl:ID>
16     </cprovl:target>
17    </cprovl:IF>

18 <!-- ****************  XACML generated equivalent policy - up to target statements *************************-->
19 <Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd- 17" ...xacml-core-v3-schema-wd-17.xsd"
20    PolicyId="...ex:policy1" Version="1.0" RuleCombiningAlgId="...rule-combining-algorithm:first-applicable">
21   <Description/>
22    <PolicyDefaults>
23    <XPathVersion>http://www.w3.org/TR/1999/Rec- xpath-19991116</XPathVersion>
24    <Rule RuleId="urn:oasis:names:tc:xacml:3.0:ex:rule1" Effect="Permit">
25     <Description>Re-sharing of data is not permitted by contributors </Description>
26      <Target>
27       <AnyOf>
28        <AllOf>
29         <Match MatchId=".xpath-provenance-d-id-match">
30          <AttributeValue XPathCategory="..attribute-category:resource"
31           DataType="...data-type:xpathExpression" AttributeId="...xacml:3.0:resource-id">r:e001
32          </AttributeValue>
33          <AttributeDesignator MustBePresent="false" AttributeId="...xacml:3.0:resource-id"
34           DataType="...3.0:data-type:xpathExpression" Category="...3.0:attribute-category:resource"/>
35         </Match>
36        </AllOf>
37       </AnyOf>
38       <AnyOf>
39        <AllOf>
40         <Match MatchId="...xpath-provenance-d-id-match">
41          <AttributeValue XPathCategory="..subject-category:access-subject"
42           DataType="...data-type:xpathExpression" AttributeId="...xacml:3.0:subject-id">r:ag001
43          </AttributeValue>
44          <AttributeDesignator MustBePresent="false" AttributeId="...xacml:3.0:subject-id"
45           DataType="..data-type:xpathExpression" Category="...subject-category:access-subject"/>
46         </Match>
47        </AllOf>
48       </AnyOf>
49      <AnyOf>
50       <AllOf>
51        <Match MatchId="...xpath-provenance-id-match">
52         <AttributeValue XPathCategory="...cloudprovenance:cprovl:1.0:attribute-category:cprov"
53          DataType="..data-type:xpathExpression">ex:a-share</AttributeValue>
54         <AttributeDesignator MustBePresent="false" AttributeId="..xacml:3.0:action-id"
55           DataType="...:data-type:xpathExpression" Category="...attribute-category:action''/>
56        </Match>
57       </AllOf>
58      </AnyOf>
59     </Target>
```

FIGURE 6.15: cProvl Policy and XACML Policy - up to Target Statements

defined at the target. This edge can be read as: check to see if requested user (its value is also referenced from the variable (ex:ag001)) is the originator of the resource. The second and third statements check to see if the "share" process uses the resource, which was initially called by the user. If all true, then he/she is permitted to share, otherwise denied.

Lines 108-124 are the comparable conditional statements plus the rest of the policy in XACML (for the lines 18-51 of the cProvl policy). The conditional statements are more complex than the target statements, since they use grouping, conditional and logical operators. XACML provides a good support for conditional and logical operators and we have leveraged on them (line 30). However, for the validation of the conditional statements against the traceability store, we can use the function called labs:orange:com:function:ext:xpath-provenance-node-match (see Section 6.4.1.3) (lines

```
27        <!-- Convert cProvl conditional statements  into XACML conditional statements -->
28        <xsl:if test="count(cprovl:SUCH-THAT)!=0">
29         <Condition> <xsl:for-each select= "cprovl:SUCH-THAT/cprovl:statements">
30          <xsl:choose> <xsl:when test="count(cprovl:grouping)=0">
31           <Apply
32            FunctionId="..tc:xacml:1.0:function:and">
33             <xsl:for-each select="cprovl:statement">..
34              <Apply FunctionId= "..xpath-provenance-node-match">
35               <AttributeValue DataType="..data-type:xpathExpression">
36                 <xsl:value-of select="child::node()/@prov:id"/>
37               </AttributeValue> </Apply> </Apply>
38             </xsl:for-each>
39              </Apply></xsl:when>
40          <xsl:otherwise>
41          <xsl:apply-templates select="cprovl:grouping"
42            mode="loop"/> </xsl:otherwise> </xsl:choose>
43       </xsl:for-each> </Condition> </xsl:if>
44      </Rule>  deny all </Rule>
45    </Policy> </xsl:template> ...
46 </xsl:stylesheet>
```

FIGURE 6.16: cProvl Policy to XACML Policy Stylesheet - From the Conditional Statements to End of Policy

109,114 and 119), which adds the traceability support for conditional statements.

6.4.3 Traceability Store

The traceability store is designed for persistent storage of XML-based traceability records. Figure 6.18 shows the hierarchical structure of the traceability store which is implemented using the open source 'eXist DB' [130] (referenced to layers 4 & 5 on the server side stack, see Figure 6.2). The XML database is based on Java, and can run on multiple platforms. It organises each entry as either a record or a collection as a hierarchical structure. A collection can have one or more records and collections iteratively.

The base collections are the"traceabilityStore" and "policyStore". Each store contains collections for one or more services, in our case it is the ConfidenShare service. The ConfidenShare service collection consists of two additional collections, namely "service-Traceability" and "policyTraceability". These two collections are responsible for storing all the service-related traceability and policy-related traceability records. The PolicyStore on the other hand is designed to store all the policies.

Figure 6.19 shows an eXist reference to a traceability record (TrDoc1.xml) in the TraceabilityStore. The content of the file are traceability entries containing a cResource and an Agent. The cProv traceability record can be directly mapped to a Prov record. We created a XSLT stylesheet (see Section A.4) that transforms a cProv traceability record into Prov record. This enables us to take advantage of existing tools for Prov, such as the validator [149].

For advance XML file manipulation, eXist has support for XQuery [131], with its own custom implementation of insert, update, and modify of XML nodes; features which are extensively used by the traceability store API.

The traceability store updates a record by adding the new nodes at the end of the record.

```
<!-- ****************  cProvl policy - From the conditional statements to end of policy ************************-->
  <cprovl:SUCH-THAT>
   <cprovl:statements>
    <cprovl:statement>
     <cprov:hadOwnership prov:id="ex:hd1"> <prov:entity prov:ref="r:e001"/> <prov:agent prov:ref="r:ag001"/>
      <cprov:ownershipType>cprovd:Originator</cprov:ownershipType>
      </cprov:hadOwnership>
     </cprovl:statement>
    <cprovl:statement>
       <prov:used prov:id="ex:u005"> <prov:activity prov:ref="r:a-share" /> <prov:entity prov:ref="r:e001" />
      </prov:used>
     </cprovl:statement>
     <cprovl:statement>
      <cprov:wasInitiallyCalledBy prov:id="ex:wicb001"> <prov:activity prov:ref="ex:a-share" /> <prov:agent prov:ref="r:ag001" />
       <cprov:accessMedium>cprovd:Laptop</cprov:accessMedium> <cprov:accessNetwork>cprovd:wifi</cprov:accessNetwork>
     </cprov:wasInitiallyCalledBy>
    </cprovl:statement>
   </cprovl:statements>
  </cprovl:SUCH-THAT>
  <cprovl:THEN>
   <cprovl:var cprovl:var_type="cprovd:new" cprovl:var_identifier="ex:result" />
   <cprovl:entity prov:id="ex:result"> <cprovl:actionType>cprovd:Permit</cprovl:actionType>
    <cprovl:resourceType>r:processRef</cprovl:resourceType>
   </cprovl:entity>
   <cprov:hadOwnership>
    <prov:entity prov:ref="ex:result" /> <prov:agent prov:ref="ex:ag001" />
     <cprov:ownershipType>cprovd:Possession</cprov:ownershipType>
   </cprov:hadOwnership>
  </cprovl:THEN>
 </cprovl:RULE>

<cprovl:RULE>
  <cprovl:entity> <cprovl:description>Otherwise deny all</cprovl:description> </cprovl:entity>
  <cprovl:DEF>
   <cprovl:entity><cprovl:range>cprovd:All</cprovl:range> </cprovl:entity>
  </cprovl:DEF>
  <cprovl:THEN>
   <cprovl:var cprovl:var_type="cprovd:new" cprovl:var_identifier="ex:result" />
   <cprovl:entity prov:id="ex:result"> <cprovl:actionType>cprovd:Deny</cprovl:actionType>
    <cprovl:resourceType>r:processRef</cprovl:resourceType>
   </cprovl:entity>
   <cprov:hadOwnership> <prov:entity prov:ref="ex:result" /> <prov:agent prov:ref="ex:ag001" />
    <cprov:ownershipType>cprovd:Possession</cprov:ownershipType>
   </cprov:hadOwnership>
  </cprovl:THEN>
 </cprovl:RULE>
</cprovl:POLICY>

<!-- *************  XACML generated equivalent policy - from conditional statements to end of policy ***************-->
  <Condition>
   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
    <Apply FunctionId="labs:orange:com:function:ext:xpath-provenance-node-match">
     <AttributeValue DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
         XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:cprov">ex:hd1
     </AttributeValue>
    </Apply>
    <Apply FunctionId="labs:orange:com:function:ext:xpath-provenance-node-match">
     <AttributeValue DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
         XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:cprov">ex:u005
     </AttributeValue>
     </Apply>
      <Apply FunctionId="labs:orange:com:function:ext:xpath-provenance-node-match">
       <AttributeValue DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
            XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:cprov">ex:wicb001
       </AttributeValue>
      </Apply>
   </Apply>
  </Condition>
 </Rule>
 <Rule RuleId="urn:oasis:names:tc:xacml:3.0:" Effect="Deny">
   <Description>Otherwise deny all</Description>
 </Rule>
</Policy>
```

FIGURE 6.17: cProvl Policy and XACML Policy - From the Conditional Statements to End of Policy

The process is relatively fast, and can handle reasonably large number of Prov entries. However, processing a single record with large entries can be challenging, in terms of accessing, querying and updating. We therefore chose to limit the number of entries per record, and spread the entries through multiple records. This ensures that the records are manageable and guarantee a reasonable performance time.

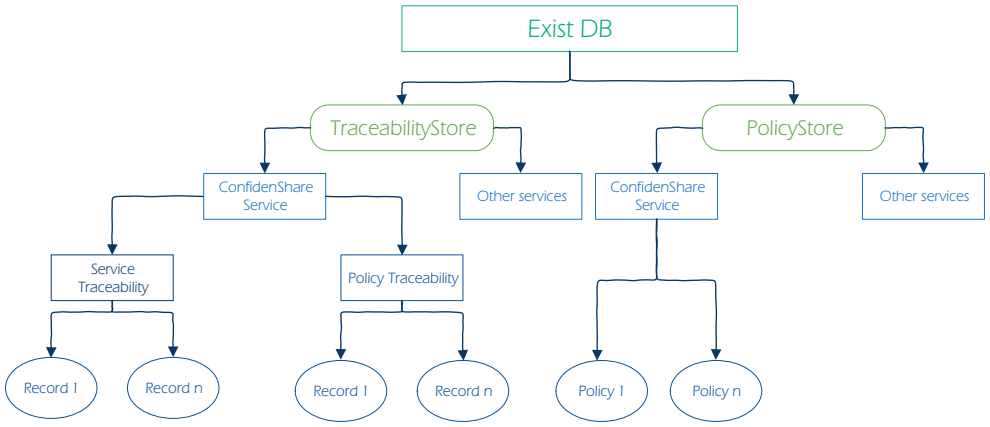


FIGURE 6.18: Traceability Store Structure

```
1    exist:/db/cProv-ProvenanceStore/ConfidenShare/ServiceTraceability/TrDoc1.xml

2  <cprov:traceabilityDocument>
3      <cprov:cResource prov:id="ex:e-provrecord">
4          <cprov:userCloudRef>=bob[=]!:uuid:f81d4fae</cprov:userCloudRef>
5          <cprov:vResourceRef>[=]!:pid:g81d4fde-record</cprov:vResourceRef>
6          <cprov:pResourceRef>192.168.1.34</cprov:pResourceRef>
7          <cprov:isReplicable>true</cprov:isReplicable>
8      </cprov:cResource>
9      <prov:agent prov:id="ex:ag00676762">
10         <prov:label> Test </prov:label>
11     </prov:agent>
12 </cprov:traceabilityDocument>
```

FIGURE 6.19: Traceability Record

6.5 Framework Service Integration

We have successfully integrated the framework with the ConfidenShare service (section 2.1). The service is able to generate and store its traceability data, and apply traceability-based enforcements.

The sequence diagram (Figure 6.20) demonstrates the interactions between the framework's components and the service. It shows a user, Bob, invoking a resource share request on the ConfidenShare web client (line 1-3). The client (using cProv client API) generates provenance data for this invocation and interacts with the 'ProvenanceHandler' for translating it to XML Prov elements, then storing it using the cProv Store API (line 4-8).

An example of 'cProv client API' (layer 1 of the client stack, see Figure 6.1) invocation is as follows:

```
provenanceStatementGenerator.cResource("e-" + filename, "text", + httpAddress + "/" +
filename, resourceID,"", true,"","");
```

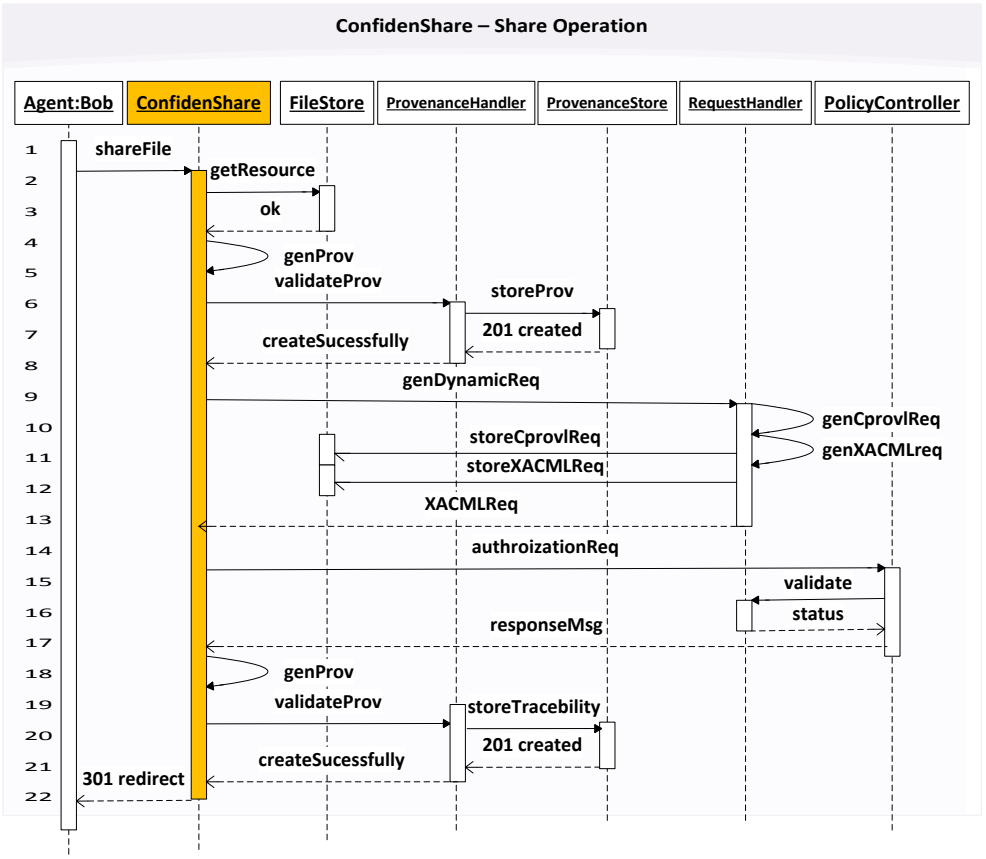


FIGURE 6.20: Framework Integration with the ConfidenShare Service

This invocation generates an XML encoded 'cprov:cResource' traceability entry. The entry is then sent to the messaging queue to be stored on the server. The cProv Store API is invoked to store the entry in a traceability store.

```
REQUEST:  POST http://labs.orange.com/uk/cloudProvenance/cprov-ProvenanceStore/
    traceabilityDocument/ConfidenShare/ServiceTraceability HTTP/1.1
    Host: labs.orange.com   Content-Type: application/xml Content-Length: nnn
         ....
          <Entity prov:id='confidenshare:e-resourceX'> ...
          </Entity/>
          ....
RESPONSE:  HTTP/1.1 201 Created
   <cprov:TraceabilityDocument>
      <cprov:cResource prov:id='ex:confidenShareRecord1> ...</cprov:cResource>
</cprov:TraceabilityDocument>
```

This example shows that a generated entity statement (e-resourceX) is ready to be posted for permanent storage. This process is handled seamlessly by the cProv Store API. It is creating a traceability record 'confidenShareRecord1' with the element 'e-resourceX' by appending it to the end. If it is successful, a response code of 201 is returned with the record Id (confidenShareRecord1).

The next sequence (line 9) on the diagram is the ConfidenShare service generating and initiating a request (using the cProvl Client API) to validate against the service requirements for compliance (as defined in Section 2.2). The policy controller executes the request using the defined cProvl policy (Section 2.2) in the XACML engine (lines 14-16). If the response is granted, then the resource share is permitted, and the provenance record is updated (lines 17-22).

An example of a dynamic request using the cProvl Client API) (see Section A.9 for the API interface in Java) for a share request is as follows:

```
    // service provenance-based control request integration
    dpr.constructRequest(
        session.get(SESSION_USER_NAME), // get session username
       filename.getName(), //name of filename to be shared
       getShareProcessName(),   //name of the share process
       getEnvironmentValue() // reference of the environment
  );
//generates a cProvl request (see below)
```

This example can be read as a 'ConfidenShare' session user ('Bob') is requesting for authorization to share a file (document1). This request gets automatically translated into cProvl request, as follows.

```
    <cprovl:PolicyRequest ....> <cprovl:Agent isRef="false" prov:id="confidenshare:ag-Bob"/>
        <cprovl:Entity prov:id="confidenshare:document1">
            <cprovl:reqField>cprovd:Resource</cprovl:reqField>
            <cprovl:fieldValue isRef="false">confidenshare:document1 </cprovl:fieldValue>
        </cprovl:Entity> ...
    </cprovl:PolicyRequest>
```

An XACML equivalent of this request is as follows.

```
<Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17  ... CombinedDecision="false">
   <Attributes Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
      <Attribute ... AttributeId="urn:oasis:names:tc:xacml:3.0:subject-id">
         <AttributeValue DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
         XPathCategory="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">ex:ag-Bob
         </AttributeValue> </Attribute> ...
   </Attributes>...
 </Request>
```

This request is used by the extended XACML engine to determine if it is compliant with the defined policies (Section 2.2) using the ConfidenShare service's traceability data.

Once a response is sent from the PolicyController to the ConfidenShare (line 17-20), the response is validated, and traceability of the outcome is stored. Based on the outcome, the ConfidenShare redirects to the appropriate response page (e.g. granted/denied).

6.6 Discussion

The traceability framework presented in this chapter is designed to integrate with an existing service easily by using sets of APIs that provides functionalities such as traceability declaration, generation, storage, and policy based enforcement. The framework provides client and server-side stacks which handle all underlying complexities, and provides the necessary backbone infrastructure that is required to enable full integration of traceability support.

Much of the APIs provided by the framework are designed to be called internally within and along with the service. While this provides fast connectivity, and good level of performance, it is not secure. The calls content are not encrypted or called over a secure channel. For additional security, the content sent to the server can be encrypted and IPSec [69] or SSL [183] may be considered, however, this may be at the expense of performance, and potentially scalability.

XML is currently the preferred method of encoding, as it is expressive, and can handle complex modelling, but at the same time it is also a relatively heavy weight solution. While our service scenario (Section 2.1) is more than sufficient to be handled by XML, a highly transactional distributed service with a very large traceability data generation, may require a non-XML based traceability store (no-Sql DB) with alternative traceability representation, such as JSON may be preferred.

The XML structure of the traceability document inherits from the Prov-XML, however, the cProv XML structure uses its own elements and edge names that may not be recognized directly by Prov-XML compliant implementations. We have created an XSLT translator that provides full mapping to Prov-XML. This would allow us to leverage on and use Prov tools and other Prov compatible services.

The mapping of cProvl policy language to XACML proved to be quite challenging. Mapping of single (text) value of XACML policy statements to multiple values (XML) of cProvl policy statements, and validating them against traceability statements (XML) required introducing new functions (see Section 6.4.1) to handle them. The implementation and integration of these functions with the Balana Engine is achieved by the support of the Spring Framework [100], which minimizes the integration dependency and reduces to a small number of lines of modifications. This results in stable running of cProvl polices in the XACML-based (Balana) engine and produces output which are reliable. However, due to relatively complex architecture (see Figure 6.3), more monitoring and validations may be required for commercial environments.

The Spring framework is also used with the client stack of the our traceability framework to integrate with the ConfidenShare service. With the support of the dependency injection of Spring, this can minimize the overall integration effort required by the developers and can focus more on the implementation of business logic.

6.7 Summary

In this chapter we have presented a traceability framework that is able to integrate traceability capturing capability, as well as policy-based enforcement mechanism. The core features of the framework are: firstly, it provides high level API for ease of integration with a service. Secondly, it leverages on industrial standards such as XML, XACML and Prov to minimize the adoption and integration complexities. Thirdly, it allows generation and capturing of traceability data dynamically. Finally, using this data, it facilitates compliance based enforcement.

The framework has been successfully integrated with the ConfidenShare service, and is able to generate traceability data and permanent storage of it in the traceability store. The write-only traceability data growth can lead to a very large dataset. Such volume of data can be managed by using the TTL property (relevance of data) of traceability data, which would archive and store any traceability data considered no longer relevant separately. The service is also able to exercise compliance based enforcement using traceability data for some of its operations.

Chapter 7

Evaluation

In the previous chapters, we detailed the 'ConfidenShare' service and its requirements. Using the extended provenance model (cProv) and policy language (cProvl), a policy enforcement framework implementation was presented. In this chapter, we evaluate the fulfilments of these requirements, the framework and its integration with the service, and demonstrate the real-world applicability of our conceptual design.

We have conducted our evaluation in a number of stages. First, we present the service requirements by modelling the service level provenance data, policies, requests and validating them against the policy enforcement system.

Second, we analyze the performance of the framework in three ways:

- **Traceability Store** - We analyze the performance of the store in a controlled environment, with two variants of traceability data entries. First, creating and storing of 1m traceability entries in multiple of 10 statements, followed by 1m entries in multiple of 20 statements.
- **Policy Language** - We determine the performance of policy target and condition statements by incrementally adding new statements (target & condition) and measuring the execution time.
- **Policy Enforcement** - We examine the performance of the policy enforcement by measuring the cumulative time for the end-to-end execution of a policy. This includes the generation of traceability data, policy requests, translations and executions in a policy engine. A total of 1m executions is recorded.

Finally, we demonstrate the integration of the traceability framework with the ConfidenShare service by applying some of the service requirements and determining potential overheads.

7.1 Framework Performance

The framework consists of three main parts: the traceability store, policy language, and the policy enforcement. All combined together to achieve end-to-end service level requirements for the 'ConfidenShare' service. It is important to conduct performance on each part to identify any potential performance issues that can result in dissatisfaction of the service usage.

7.1.1 Benchmark Environment

The experiments used to evaluate the performance of the framework is based an Intel (R) Core (TM) i7-2820QM CPU @2.30 GHZ, with 6Gb of RAM and 600Gb of disk space.

7.1.2 Traceability Store

Hypothesis 1 (Service Statements)**.** *The integration of the cProv traceability model with the 'ConfidenShare' service generates and stores traceability data without major impact on the running of the service.*

Method A - Online processing: We generate and store the traceability statements using the cProv client API, and cProv REST API. For online processing, we are using the 10 traceability statements (statements for validating Policy one (see Section 2.2.1)). To generate the traceability statements, each statement is first represented as a Java object, which is then marshalled to an XML element using JAXB (shown in Figure 7.1). This is then sent to the queue to be stored in the TraceabilityStore. The connector collects one statement at a time from the queue and waits for it to be processed before collecting another one (hence synchronous communication between connector and remote storage). The TraceabilityStore uses multiple files to store the elements. It limits each file to a maximum of 10,000 entries.

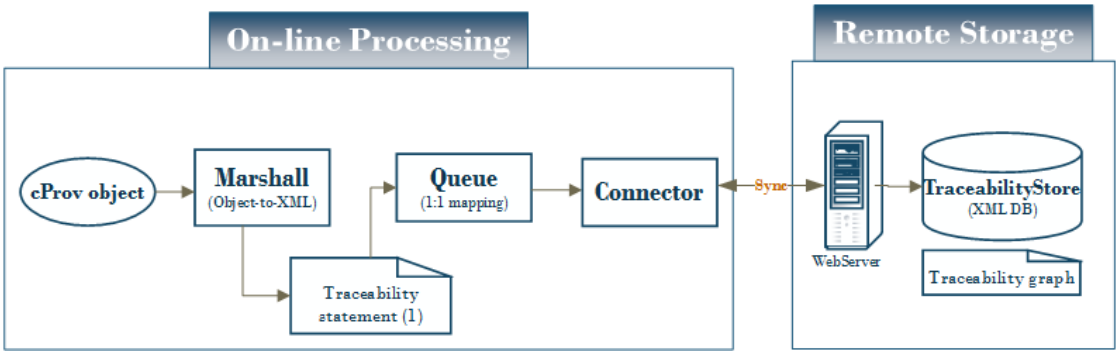


FIGURE 7.1: Online Processing Structure

This process is repeated 100,000 times; generating a traceability graph of 1 million entries. The time it takes between creating and storing of a statement is recorded as a unit of 10 statements resulting in a total of 100,000 measurements.

Method B - Batch processing: cProv client API and cProv REST API are also used to generate and store the traceability statements. However, in the batch mode, each of the Java object statements is collated in a cluster of 10 for the policy one (batch-10) and 20 for the policy two (batch-20) (see Figure 7.2). This is then marshalled into a larger XML document containing all the statements, then sent to the queue to be stored in the store.

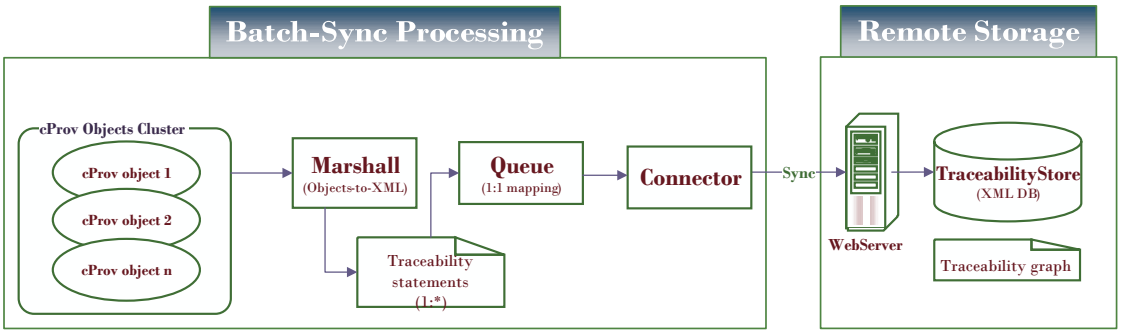


FIGURE 7.2: Batch Processing Structure

This process is repeated 100,000 times for graph one, and 50,000 for graph two, and added to the existing traceability graphs. This produces two graphs of 1 million statements respectively. The time it takes between the creating and storage of statements are recorded as a unit of 10 statements in the first graph and 20 in the second (resulting in 2000 measurements).

Analysis - Figure 7.3 presents the results of the three tests (batch-10, batch-20 and online). Each test plotted represents 100 results (each result contains a mean value with error bar at the 95% confident interval). For batch-10 and online, each result represents an average time for 1000 measurements (each measurement contains 10 statements). This results in measurements of a total of 1m statements (100 results * 1000 measurements * 10 statements per measurements). For batch-20, each result contains an average time for 500 measurements (each measurement have 20 statements). This results in measurements of a total of 1m statements (100 tests * 500 measurements * 20 statements per measurements).

Overall, the graph shows a good correlation between the traceability entries (generation & insertion) and the time. As expected, for the online processing, 10 statements per unit took on average of 0.559s to complete. In batch-processing, the average unit time was much quicker of 0.083s per unit for batch-20, and 0.0552 for batch-10.

The batch processing is almost 7 times faster than online processing. However, what is

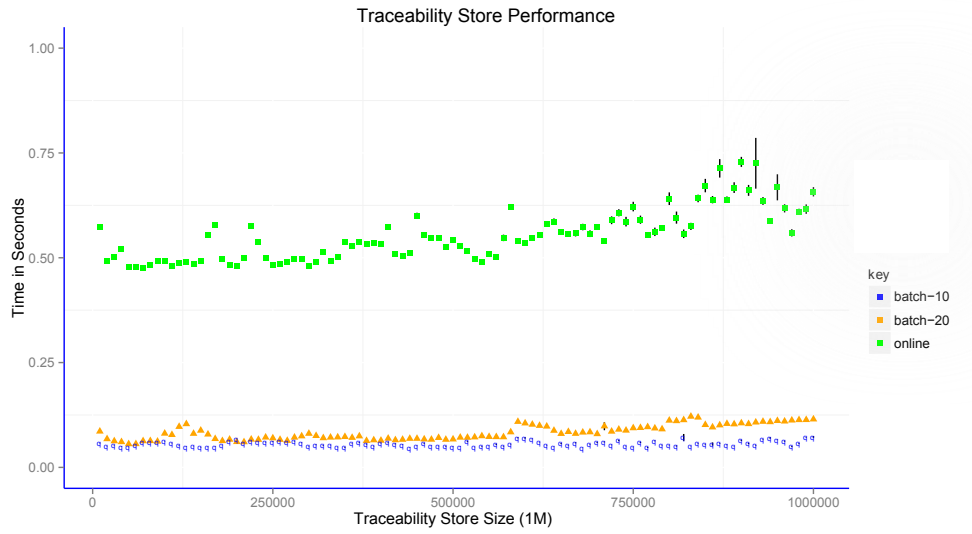


FIGURE 7.3: Traceability Store Size Vs Time for 1M cProv Statements

interesting is, batch-20's average time is almost twice as the batch-10s, but if we take the overall time, then it is 25% faster than batch-10. Therefore there is a trade-off between the initial batch size and total entries.

Table 7.1 shows the summary results with the standard error, standard deviation and confidence interval. For each entry we can see a very small error, which indicates the result is relatively consistent. However, on the Figure 7.3 we can see some of the online plots (top two series) contains relative large error bars in comparison to the batch processing (bottom two series, due to very small errors, the error bars are not visible), which suggest online processing is not as consistent as batch-processing due to higher rate of markshalling of traceability statements and Web connectivity).

Results Summary						
Entry	N	Average Time(s)	SD(s)	SE(s)	CI(s)	Total Statements (s)
batch-10	1m	0.0559	0.0380	0.000120	0.000236	5587.374
batch-20	1m	0.083	0.0494	0.000220	0.000433	4158.693
online	1m	0.552	0.312	0.000987	0.00193	55243.97

TABLE 7.1: Summary Results of Traceability Store

- N - number of items (prov unit)
- SD - Standard deviation (s)
- SE - Standard error (s)
- CI - Confidence interval (set to 95%)

While these results demonstrate the superior performance of the batch-mode, there is a trade-off between the performance and resource usage. The larger the size of the batch,

the more resource intensive (memory and CPU) it is likely to be, with a greater network load (higher bandwidth), hence better performance. The smaller the batch size, the less resource intensive, but with a greater network overhead (network calls) therefore compromised performance.

The marshalling between Java objects and XML is relatively expensive; the majority of the overhead costs (2/3) of online processing is related to this process. Other alternative technologies need to be considered or developed for improved performance of online mode. The overall impact on the production system would not be significant, since the marshalling process would only be performed for a group of provenance statements (i.e. execution of an event) and the storage would take place asynchronously.

7.1.3 Policy Language

Hypothesis 2 (Policy Statements)**.** *The number of statements within a policy determines the execution time. With the addition of new statements, execution time increases linearly (condition statements should take longer to execute than the target statements of the policy) with the size of a policy.*

Method - Policy one (see Section 2.2.1) contains four targets and three condition statements. *A new policy statement (resource related) is added incrementally for each run* to the existing policy per execution. This process is repeated 100 times, first with the condition statements, and then with the target statements. The time it takes to execute a policy, from the request to the response (excluding the policy update time), is recorded. A total of 200 measurements (100 target statements and 100 condition statements) are collected.

Analysis - As it can be seen from Figure 7.4, there are a total of 25 results (each result represents a mean value with error bar at the 95% confident interval) for conditional and target statements each. Each result represents average time of 4 measurements. This equates to a total measurements of 100 statements (4 measurements * 25 results).

With the addition of a new policy statement for each run, there is proportional increase in the time (on average 0.0495s for conditional statement and 0.0444s for target statements) it takes to execute the policy; which is relatively linear.

The condition statements take longer to execute than the target statements, this is as expected since condition statements are multi-valued and contain dynamic variable references; hence are more complex than single valued target statements.

From Table 7.2, we can see a policy with 100 condition statements takes almost 2.60s to execute, while 2.37s for target statements. While this may seem relatively expensive in

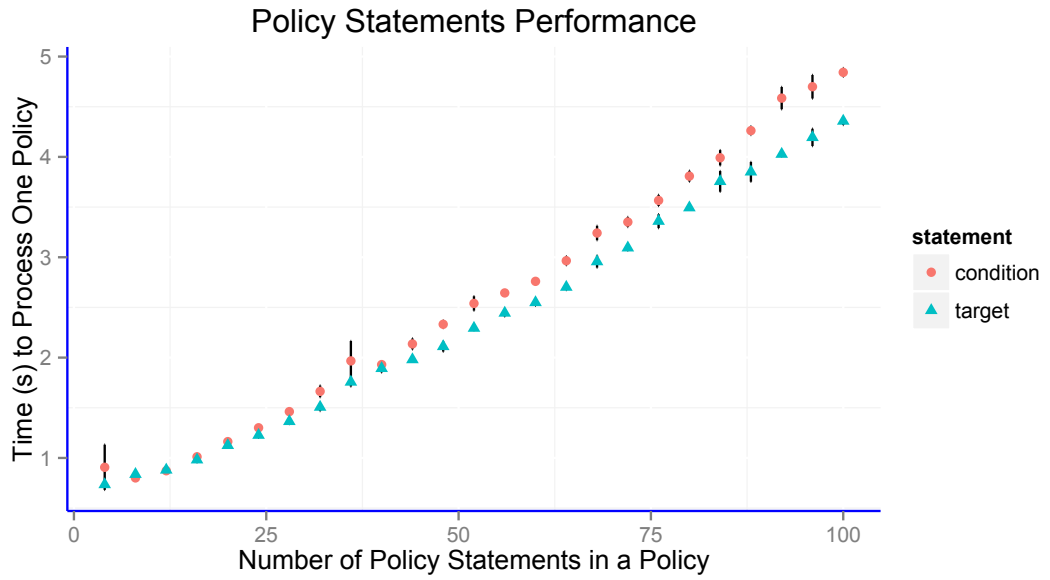


FIGURE 7.4: Policy Statements Size Vs Time for One Policy

Results Summary						
Entry	N	Average Time(s)	SD(s)	SE(s)	CI(s)	Total Time (s)
condition	100	2.592	1.283	0.128	0.255	259.215
target	100	2.379	1.140	0.114	0.226	237.912

TABLE 7.2: Summary Results of Policy Language

terms of time, in a commercial service, it is unlikely to reach anywhere near this size. We believe, the policy would be too complex to administer, manage and maintain. Based on experience, the more realistic case would be on average between 5-20 statements.

This result indicates policies are, in theory, scalable and can handle large complex business policy requirements. However, if a policy contains a large number of statements, then optimization may be required to obtain better performance. For example, splitting a large policy into multiple small policies and running non-dependent policies simultaneously.

In a production system, based on our service requirements (see Section 2.2), the policies are relatively small and contain modest number of statements (see Appendix A.7), hence the overall performance would not be significantly compromised.

7.1.4 Policy Enforcement

Hypothesis 3 (Policy Enforcement)**.** *The translation of cProvl policies into an XACML policies for enforcement are semantically equivalent. The addition of policy enforcement adds overhead costs relative to the number of time the policy is run by the policy execution*

mechanism.

Method - We use the static cProvl policies one and two of the ConfidenShare service (see Section 2.2.1). The requests for policies are generated dynamically using cProvl client API, which are then translated into an XACML equivalent and executed in a extended XACML engine. The engine uses traceability data obtained based on the previous method to evaluate each policy.

This process is repeated 1000 times. We record four variations of results, the first execution of policy one with and without the provenance time, and secondly the same with policy two. The start/finish times are recorded for each variant.

Analysis - Figure 7.5 shows four items (policy1, policy 1 with prov, policy 2 and policy 2 with prov). Each item plotted represents 100 results (each result represents a mean value with error bar at the 95% confident interval). Each result contains an average time for 10 measurements. This results in measurements of a total of 1K policies (10 policies * 100 measurements). We can see the execution time for policy one, is an average time of 0.425s (0.483s with traceability generation/storage time) per execution, and for policy two it took 0.519s (0.59s with cProv).

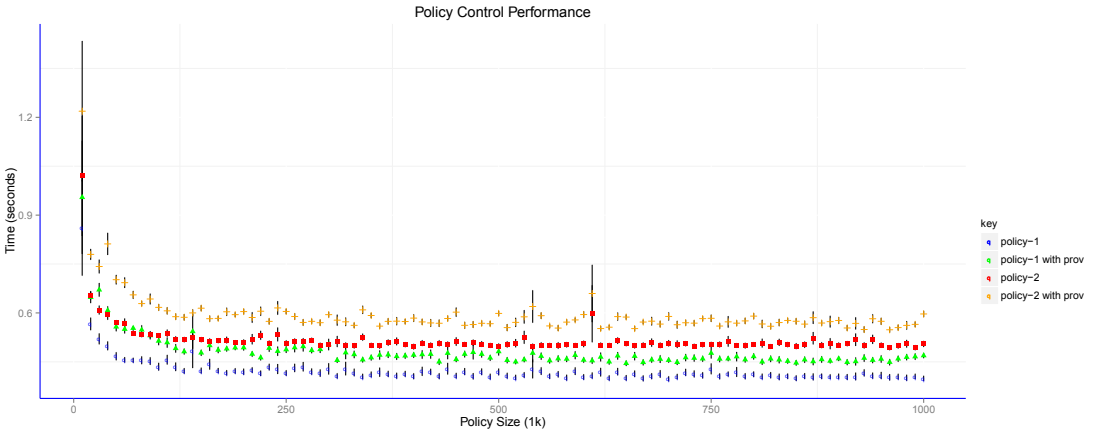


FIGURE 7.5: Policy Execution Vs Time for 1K Policies

There is a relatively high execution time at the first few policies. We believe this may be due to several factors such as: fragmented hard disk, too many background program running at the same time, class loader initialization of classes, loading OS level resources, and others. The remainder of the graph shows relatively consistent policy execution verses time.

The standard deviation (see Table 7.3) for both policies are very small, which indicates, they are fairly close to the average time.

This result indicates, the additional of provenance-based policy enforcement would add minimal overhead (around 1/2 a second) to the service case. This would not have a

Results Summary						
Entry	N	Average Time(s)	SD(s)	SE(s)	CI(s)	Total Time (s)
policy-1	1k	0.425	0.0767	0.00242	0.000236	-
policy-1+prov	1k	0.483	0.0888	0.00281	0.00551	-
policy-2	1k	0.519	0.0888	0.00281	0.00551	-
policy-2+prov	1k	0.596	0.110	0.00348	0.00682	-

TABLE 7.3: Summary Results of Policy Enforcement

tangible impact on the usability of the service, since it only applies to certain control gates on the application. This can be very much analogous to the security control we see on the banking application, where on average it requires few seconds to validate and process payment related requests.

Overall, the framework performance is relatively consistent, and in theory it can be scaled to meet the demand of the service requirements. The TraceabilityStore can handle a million records (with partitioning), without having significant performance degenerations. It can be scaled to handle millions of statements by adding additional hardware. The increase in the number of provenance statements can also scale linearly, but we do not envision any services requiring such large numbers of statements.

7.2 Service Integration

In this section we evaluate the integration of the traceability model and policy language, and the overhead it adds to the ConfidenShare service. In order to do this, we evaluate the performance in two stages. First we look at the cost of adding provenance statements to the registering of a user and creating of a resource. Secondly, we look at the cost of sharing a resource via the integrated provenance-based compliance enforcement.

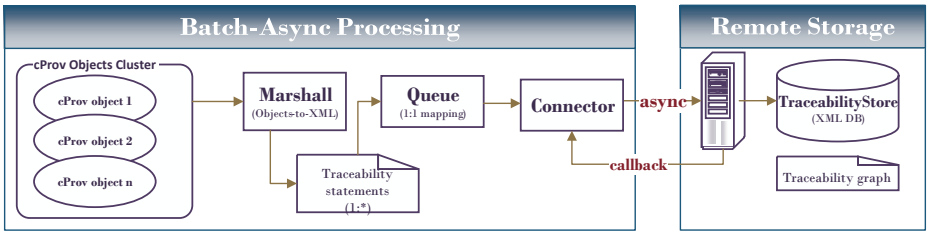


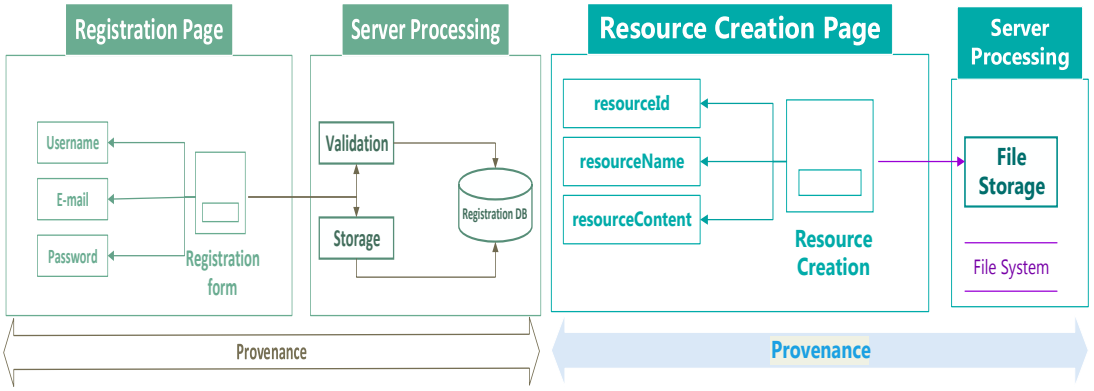
FIGURE 7.6: Async Batch Processing Structure

We have chosen to use the batch-process mode for integrating provenance with the ConfidenShare service (see Figure 7.6) based on asynchronous communication. This allows the service to run the provenance integration in a non-blocking mode.

7.2.1 Creation of a User and Resource

The creation of a user necessitates entering of a username, email and password on the registration form (see Figure 7.7 left). This information is first validated against the database for duplications. If the user record is not found, an entry in the DB is created, followed by a web-session for the user.

A resource, typically represents textual information, which can be of type: meeting minutes, memos, and others in a file. The content of the resource is created via the Web interface (see Figure 7.7 right), where a user enters the name, title and the content of the resource. The system creates a file with the content and stores it on the WebServer's local drive. The full provenance of the process is captured.



(a) User Registration (b) Resource Creation

FIGURE 7.7: User Registration and Resource Creation Process

The provenance trace, from the beginning to end of the registration process, is recorded.

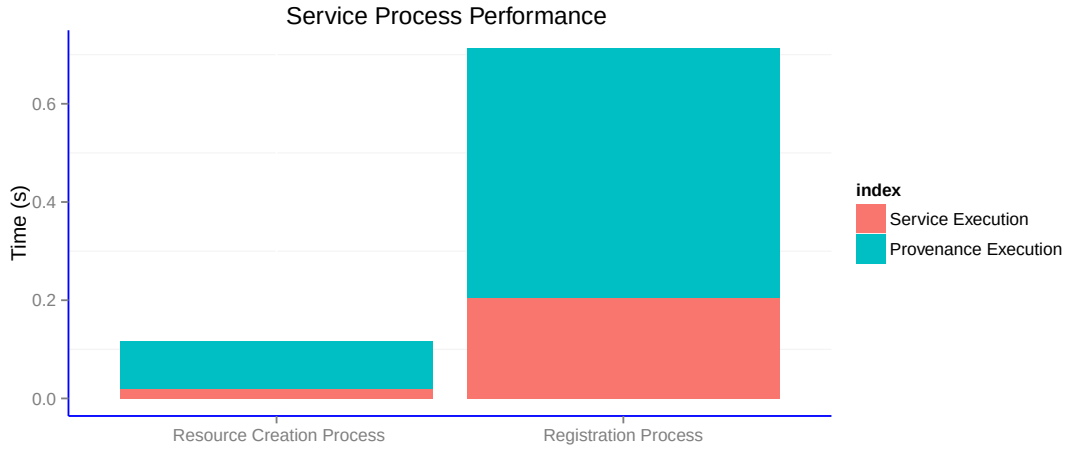
Hypothesis 4 (Registration & Resource Creation Process)**.** *The traceability statements execution time of a process increases with the complexity of the process being invoked.*

Method - The end-to-end capture of the traceability for the registration process requires a total of 14 traceability statements. The process of recording traceability is carried out in a batch mode of 14 statements per batch. The storage process is a *non-blocking asynchronous*; essentially running in the background. We measure the performance from the click of the register button to the completion of the registration. This is carried out in two stages: with and without traceability statements.

The capture of traceability for creating a resource is similar to the registration process, where the end-to-end traceability of creating the resource is captured. It is based on 10 statements per batch in an asynchronous mode. The created resource is stored on the server as a file.

Analysis - The result indicates (see Figure 7.8) on average, it requires 0.201s to register and 0.509s with the inclusion of traceability statements. This is more than two folds of

overhead increase, but relatively low in relation to the context of the service, and would have minimal effect on the user experience.



(a) Registration performance

FIGURE 7.8: Service Registration & Resource Creation Process Performance

The process of registration is more complex than creating a resource, hence the increase of time for generation of the traceability statements. The overall time for creating a resource is much faster than a registration. This is due to the resource being created locally on the file system. In a live deployment, it would be stored remotely on a file store, and may increase the processing time.

7.2.2 Policy Enforcement

The Policy enforcement mechanism is integrated with the ConfidenShare service to validate a user request in order to grant or deny the execution of the operation(s). This is done with the support of the generated service traceability data, and predefined policies. A share request is triggered by a user via the web interface, where he/she enters the resource name and recipient.

Hypothesis 5 (Policy Enforcement)**.** *An encoded cProvl request from the client results in a cProvl encoded response results from that follows multiple stages of execution, of which the policy execution is likely to have the largest execution time.*

Method - A request is composed of the resource identifier and the intended recipient created via the cProvl request API as Java objects, that are marshalled using JAXB to cProvl XML request. We measure the duration from making a share request to generation of the cProvl request.

A cProvl request is first translated into XACML equivalent, to be validated by the policy engine. The engine uses XACML mapping of cProvl policies to execute the request and

generate an XACML response.

A cProv request made by the service is mapped to a cProvl policy. The cProvl policy is executed in the Balana engine, by first being translated into an XACML policy equivalent. The engine validates the request against the traceability store and generates an XACML response.

An XACML response is translated into a cProvl response via the XSLT transformer response module.

Analysis - From the Figure 7.9, we can see the time it takes for policy execution is *0.8* which contributes to 51% of the overall execution time. However, surprisingly invoking a service policy request to generating cProvl equivalent is on *average 0.65s*, or 41% of the overall time. The main performance hit for the dynamic request appears to be the object-to-XML mapping.

The process of translating a cProvl policy and request into an XACML request is relatively fast; it requires on average *0.067s, and 0.031s respectively.*

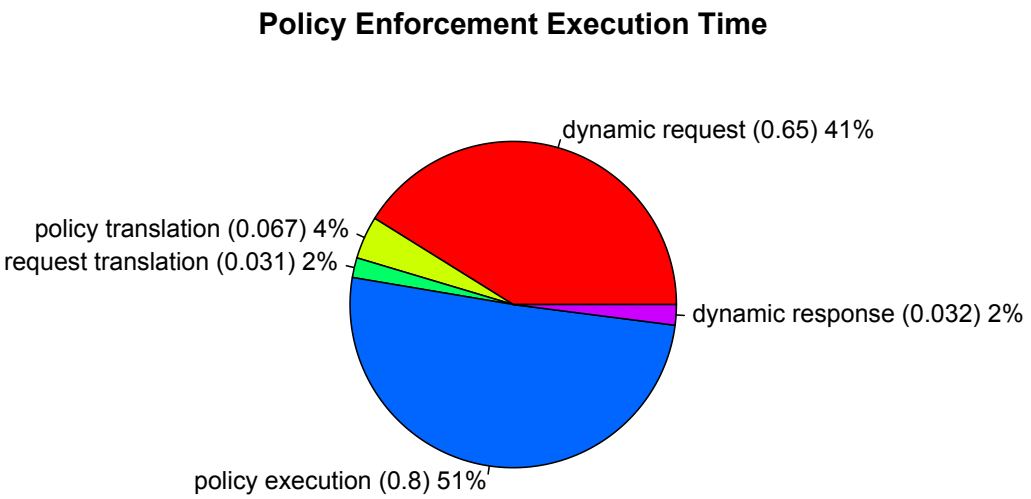


FIGURE 7.9: Policy Enforcement

We can also see on average it takes 1.5s to achieve end-to-end provenance-based enforcement, which is considered to be very reasonable for the intended ConfidenShare service. However, there are areas of improvements, firstly, JAXB appears to be very costly, and it contributes to at least 1/3 of the overall cost. A more optimized alternative solution to JAXB may help to minimize the overall cost. Secondly, we can look at ways to optimize the policies, since the amount of statements increases the processing time. At present the Balana XACML engine executes them linearly, if we can identify non-dependent

statements, then they can be run in parallel. This should also reduce the execution time.

For a commercial deployment, we would need to take into account the network lag, bandwidth, distribution of the service components, and other factors to get a true value of the overhead cost of adopting traceability-based policy enforcement. We envision in the production system, such tests would need to go one step further, with the inclusion of the end users. They would be involved in conducting trials in different conditions, such as urban and build-up cities with various latencies to get a better understanding of how the service would perform when it is deployed in a live system.

7.3 Service Requirements Satisfaction

In Section 2.2, we have identified a number of service requirements that are vital to the running of the service in a commercial environment. These requirements necessitate the enforcement needed for data sharing, processing and storing. In this section, we present the first requirement in full.

Service Requirement 1

A file (fileA) can only be shared (fileA) by a registered user (userA) to another user (userB)

This requirement ensures only registered users have full access to the service. In order to validate a user's registration status, we can use the provenance data to check if he/she has already registered.

Traceability Graph

The traceability graph in Figure 7.10 shows, a user Bob triggers the 'a-createUser' process and enter details such as the email, registration name and password. This process invokes the 'e-checkUser' process to validate the entered details, and generates a response 'e-userDoesNotExist'. The response is used by the 'a-saveUser' process, and produces a 'e-userSaved' entity.

This simple graph captures enough details to track and verify the entire registration process. We can now use this graph to validate the first requirement. A request is as follows:

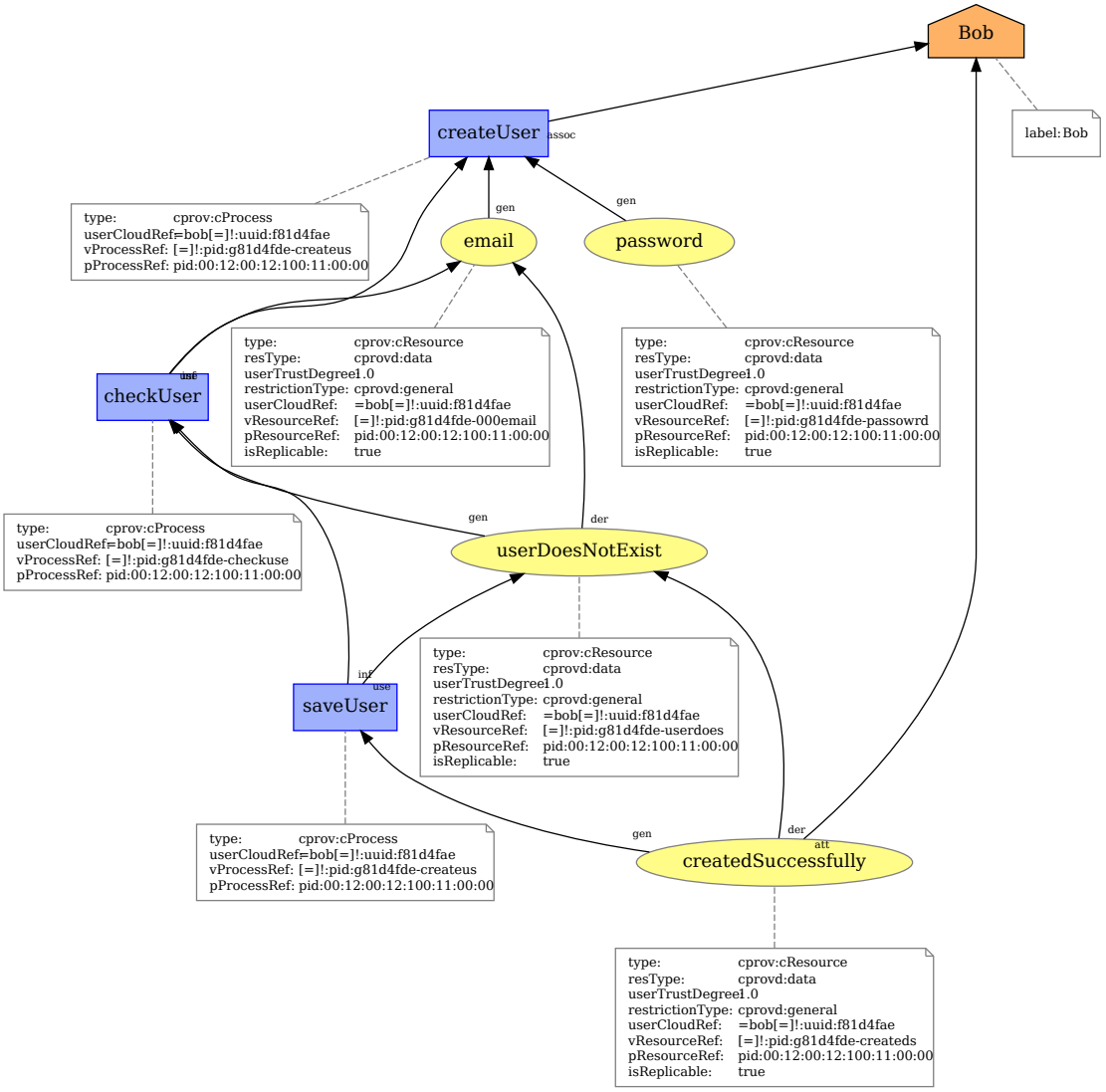


FIGURE 7.10: Requirement 1 Traceability Graph

cProvl Client Request

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<cprovl:PolicyRequest xmlns="...schema:wd-17" xmlns:prov="...prov#"
xmlns:cprovl="...cprovl#" xmlns:cprov="...cprov#">
   <cprovl:Agent isRef="true" prov:id="ex:Bob"/>
   <cprovl:Entity prov:id="ex:e-Document1">
     <cprovl:categoryType>cprovd:Resource</cprovl:categoryType>
     <cprovl:categoryValue isRef="false">ex:e-Document1</cprovl:categoryValue>
   </cprovl:Entity>
   <cprovl:Entity prov:id="ex:a-share">
     <cprovl:categoryType>cprovd:Action</cprovl:categoryType>
     <cprovl:categoryValue isRef="true">ex:a-share</cprovl:categoryValue>
    </cprovl:Entity>
</cprovl:PolicyRequest>
```

This request can be read as: A user Bob is requesting to share (via the process 'ex:a-share') the resource 'ex:Document1'. This request is validated by the following policy.

cProvl Policy

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <cprovl:POLICY xmlns:cprov="http://orangelabs.com/cprov# ..."/> ...
3      <cprovl:RULE>  ...

4          <cprovl:IF cprovl:quantifier="cprovd:There-exist">
5            <cprovl:target>
6              <cprovl:ID cprovl:var_type="cprovd:d-ref" cprovl:var_identifier="r:ag001"
7                  cprovl:category_type="cprovd:Subject">r:ag001</cprovl:ID>
8              </cprovl:target>
9               <!-- Check if process createUser exists-->
10             <cprovl:target>
11                 <cprovl:ID cprovl:category_type="cprovd:Action">confidenshare:createUser</cprovl:ID>
12             </cprovl:target>

13         </cprovl:IF>
14         <cprovl:SUCH-THAT>
15           <cprovl:statements>
16             <!-- check if there is an edge between the user ( r:ag001) and process createUser-->
17             <cprovl:statement>
18                 <cprov:wasInitiallyCalledBy prov:id="confidenshare:wicb001">
19                 <prov:activity prov:ref="confidenshare:createUser" />
20                 <prov:agent prov:ref="r:ag001" />
21                 </cprov:wasInitiallyCalledBy>
22             </cprovl:statement>

23           <!-- check the outcome if entity (ex:createdUser) has been created successfully -->
24           <cprovl:statement>
25              <cprov:hadOwnership prov:id="confidenshare:hd1">
26                 <prov:entity prov:ref="confidenshare:createdSuccessfully" />
27                 <prov:agent prov:ref="r:ag001" />
28                 <cprov:ownershipType>cprovd:originator</cprov:ownershipType>
29              </cprov:hadOwnership>
30            </cprovl:statement>
31          </cprovl:statements>
32          </cprovl:SUCH-THAT>
33           <cprovl:THEN>
34             <cprovl:entity prov:id="confidenshare:result">
35                 <cprovl:actionType>cprovd:Permit</cprovl:actionType>
36                 <cprovl:resourceId>confidenshare:share</cprovl:resourceId>
37             </cprovl:entity>
38                         ....
39                 </cprovl:THEN>
40     </cprovl:RULE>
41     <!-- deny otherwise -->
42     <cprovl:RULE>
43       ......
44     </cprovl:RULE>
45 </cprovl:POLICY>
```

The policy first checks if the target Bob and process createUser against the store (lines 6-12). If successful, the condition statements are executed. It checks (against the traceabilityStore) if Bob called the 'createUser' process and it was successful (lines 17-29).

cProvl Response

The outcome of this validation is successful and Bob is granted permission to share the resource.

The remaining requirements' (see Sections 2.2, 2.2.1, 2.2.3) validation are presented in Appendix A.7.

7.4 Analysis

The traceability-based policy enforcement integration with the ConfidenShare service has successfully facilitated the fulfilment of service level requirements. The dynamic capture of traceability data generated by the ConfidenShare service was essential in tracking the traceability of the data, and were used to validate the policies of potential compliance violations. However, first we need to analyse effectiveness of our solution with the ConfidenShare service.

The level of effectiveness of our solution can be broken down into three categories:

Accuracy - Accuracy of the policy enforcement based on the amount of traceability data needed to be captured vs time it requires.

Performance - How scalable the solution in respect to the performance.

User experience - User experience of the solution in respect to the response time.

7.4.1 Level of Details vs. Time

The current approach of capturing of traceability data is rather selective. The selection of traceability data is based on fulfilling the service requirements (see Section 2.2). Only the high level functional operations, such as method calls, user invocations, and input and output data that are required by the policy control mechanism, are captured. While it is sufficient for making adequate compliance enforcement, it does not provide an end-to-end provenance picture of the running of the service. In an event of error, or partial capture of the provenance graph, it may be insufficient or inaccurate in decision making. However, the selective capturing of traceability data, ensures it can be captured quickly and with minimal error.

7.4.2 Scalability vs. Performance

The increase in the size of the traceability entries in the store has a direct hit on scalability. The larger the individual collection of traceability entries, the less likely it can scale, and as a consequence deteriorating performance of the service. To provide greater scalability, the approach was to partition the entries in multiple graphs, with limited size. The smaller collections in multiple parts, allows greater scalability and the performance is not compromised. However, the overall size of the traceability dataset would remain the same, and may become unmanageable overtime. This can be addressed by applying a pruning strategy whereby, first identifying the non-relevance provenance data (using the TTL property of the traceability data) then removing those that do not contain any dependencies with the relevant traceability data.

The performance of the storage of the generated provenance entries are optimized by using non-blocking IO. This allows the service to continue with its operations while provenance storage is carried out in the background.

We believe to achieve a good level of scalability and performance, the ConfidenShare service should be provisioned using two or more virtual machines (VM). The first VM should host the client side interface (Web application), and second VM should host the backend components (resource repository, traceability store, policy engine and components). The architecture (see Figure 6.3) is designed to be distributed in nature, and most of the server side components can be run within their own VM if greater scalability required, however the cost needs to be considered for such provisioning.

7.4.3 User Experience vs. Response Time

The framework integration with the ConfidenShare operates in two modes, asynchronous for storing of the provenance data, and synchronous for policy enforcement. The response of the time compliance control adds up to a second which is reasonable.

7.5 Summary

In this chapter we evaluate our framework by conducting a number of benchmarks to determine their performance. First we perform the benchmark for the traceability store, followed by the policy language and finally the policy enforcement. The results show a good linear relationship between the generation and storage of traceability statements with an average of 34.3ms per statement. The integration of the policy language adds around 0.9s. Both, in theory, are scalable. In regards to policy statements, with each additional statement, the execution time increases by around 30ms.

Applying research to commercial systems at times can be challenging, because research is usually applied to theoretical context, which usually takes a simplistic view of a problem it solves and usually focuses on solving functional requirements, non-functional requirements tent to get hypothesized and end-users are not always so involved. In reality the commercial systems are complex, and the focus tents to be from the end-users perspective and requires rigorous testing to approve new solutions or functionalities, therefore, it takes a lot longer to apply theoretical research to commercial systems. However, as it can be seen with this EngD's work, it is possible to bridge the gap between the both domains by working more collaboratively and iteratively.

The overall integration of the framework with the ConfidenShare service to support policy enforcement is satisfactory and is able to support all the service requirements.

Chapter 8

Conclusions

This chapter summarises the findings of this thesis in enabling provenance-based data traceability of cloud services and providing traceability-based policy enforcements supported by a traceability framework. In order to do so, a traceability model cProv extending of W3C Prov [148] and a novel policy language cProvl (which leverages on the cProv traceability model) to model service requirements, have been defined. This overall solution integrates the framework with the ConfidenShare [168] service and satisfies its requirements as a proof of concept.

We start by providing a rationale behind this EngD, summarizing its contributions and future directions.

Telecommunication operators have been very successful in offering traditional voice and SMS services to their customers over the years. With the advancement of 3G/4G technology, that provides high data bandwidth, we have seen a proliferation of services available on the devices, primarily from the OTT (over-the-top) players such as Google, Facebook, Apple and others. While data revenues are on the increase, the traditional services are taking a big hit due to VOIP (voice over IP) and other messaging clients (such as Skype, WhatsApp and others). This decline is primarily due to OTT not having the overhead cost of running, managing or maintaining a network. Therefore, they can solely focus on services delivery at much cheaper rate than the operators and in many cases for free. This is having a big impact on the operators revenue, which is still mostly from the traditional services.

The reality is that Telecomm companies can no longer purely rely on the traditional services to maintain the operational, running and maintenance costs and still retain a healthy profit margin. It is paramount for them to evolve, and seek other potential areas of revenue. This has led Orange to explore other avenues such as cloud computing. This is seen as the next battleground for future growth. It is recognized by Orange that customers have a great deal of concerns over the trust and transparency issues related

to cloud service offerings. With the trust that operators have developed with their customers over the years, they have the necessary base to build upon with differentiating propositions. This has led to the work on the data traceability and traceability-based compliance research in this EngD.

This thesis investigates this particular research problem from two perspectives, from a commercial and technological angle. Since the EngD was sponsored by Orange, there has been an expectation that the findings of this research should lead to some form of commercial benefit and give a competitive advantage over their rivals. At the same time, to make technological advancement that could bring an added dimension to the existing technologies in use today and enable a new breed of services that are currently not available in the market.

Although the initial impetus for this research was to satisfy the business requirements for the commercial service ConfidenShare, this solution has the potential of being used by corporate and standard users. The core challenges we faced during this EngD were to maintain the balance between the commercial interests of the business and academic research. The nature of a business is that it is constantly evolving. What is relevant today may not be of an interest tomorrow. A great deal of time and effort had been spent on understanding and stabilizing the business requirements. With multiple stakeholders involved, striking the right balance proved to be challenging at times. Various expectations of stakeholders were managed by keeping all parties well informed of the current development of the work and having a common agreement of the milestones and deliverables with a single global vision of the EngD's work. Despite these challenges, we have been able to fulfil our research goals set out in the thesis. We will now briefly discuss our research contributions.

8.1 Research Contributions

From this research, there are a total of four research contributions made to the state of the art.

First Contribution - Cloud Traceability Model (cProv)

Our first contribution is related to how we can improve the transparency of a cloud service, whereby we are able to demonstrate the internal workings of a service. Having such a mechanism would greatly increase the trust of the users utilising the service. The trust is brought about by the knowledge that their data has not been accessed, modified, or shared with anyone else without their consent or awareness.

To address this, we have presented a cloud traceability model (cProv) to conceptualize running of a cloud service. The model inherits from the Prov model, which we see as the base model for generic conceptualization of a service. We have chosen the inheritance

approach as opposed to having a proprietary model. A proprietary model would give us a competitive advantage of a fully tailored solution (which would not be available to the rivals). But, this would limit the capacity of the model being compatible and able to integrate with other traceability-based services.

The main advantage of the cProv model is that it is specifically designed for modelling traceability for software, with a focus on cloud-based software services. It can be directly mapped to the W3C standard Prov. We expect in the future, support tools and technology would emerge in the market, and our model will be well placed to be used by them.

We have designed the cProv model using Prov-N [147] notation. Our feeling is that, Prov-N is good to write provenance by hand. Hence, we have chosen to use Prov-N for abstractly presenting the traceability statement. However, for implementation, XML encoding has been the preferred choice due to familiarity of the widely used technology.

It is difficult to predict how wide-scale the adoption of provenance-based traceability is likely to be in the future, however, with the aid of the W3C standard, we are hopeful that our cProv traceability model would form an important building block for future cloud services.

Second Contribution - Provenance-based Policy Language

Our second contribution is related to how can we enforce restrictions on access to resources based on the historical events. In other words, if service requirement defines certain conditions for the service users to follow, how can we check to see if he/she is abiding by these constraints, and have enforcement based on validating these constraints.

For such a requirement it is clear we need to leverage on traceability data, but also require a method to model requirements. To address this, we have created a traceability-based policy language called cProvl. The language specifically designed to provide close integration with the traceability model, and is able to model service requirements using policies and rules. A policy would typically contain one or more rules, and contains targets and conditional statements, which are based on multi-valued traceability statements.

We believe the novelty of the language is that it is one of the first provenance-based language to use extended Prov model (cProv) for defining policies for cloud based service, and uses the concept of dynamic variables to make the policies more flexible. Additionally the language retains the traceability of the request and response. The language is relatively simple to use and is expressive enough to model all the ConfidenShare service requirements.

Third Contribution - Traceability Framework

Our third contribution is related to addressing how best to integrate all the necessary components of the traceability model and policy language to operate in a harmonious way. In other words, a simple and standard way of integrating traceability and traceability-based enforcement capabilities to cloud services is required.

To address this, we have proposed a traceability framework that provides client and server side stacks to handle different functionalities required to have full end-to-end traceability-based enforcement integration. This includes, initiating, capturing and storing of traceability data, composing policy request, validation and enforcement of the request. The framework hides the underlying complexity of modelling traceability and policy enforcement mechanisms to enable smooth integration with a service via high level interfaces.

Fourth Contribution - Framework Implementation & Evaluation

Our final contribution is related to addressing the issue of how to leverage on the existing widely deployed standardized solution to benefit from features provided by the framework.

To address this, we have mapped our solution to XACML [189] by extending its core architecture. This allows cProv policies and requests to be translated into XACML equivalent to run on open-sourced Balana XACML 3.0 engine. Such mapping enhances the XACML capability by enabling it to use traceability data for policy validations.

We have successfully integrated the framework with the 'ConfidenShare' service, and have conducted a few benchmarks. The results show a good linear relationship between the generation and storage of Prov statements with an average of 34.3ms per statement. The integration of the policy language adds around 0.9s. Both, in theory, are scalable. In regards to policy statements, with each additional statement, the execution time increases by around 30ms.

To evaluate our traceability framework and its integration with the ConfidenShare service, we have conducted a number of tests. We can conclude from the benchmark results, the integration of the framework with the 'ConfidenShare', can add up to 1.5 seconds to support traceability-based enforcement, which is reasonable and encouraging. However, for a commercial deployment, we would need to take into account the network lag, bandwidth, distribution of service components, and other factors to get a true value of the overhead cost of adopting traceability-based policy enforcement.

8.2 Future Directions

Although we have achieved the primary goals of our research, there are a number of ways the work can be extended and improved upon.

Managing Growth of Traceability Data - The continuous collection of traceability data in the long run can lead to manageability issues and potential performance hits. One approach of maintaining the growth of traceability store size is by retaining only the relevant provenance data. We have introduced the concept of TTL (time-to-live), which defines how long a resource entry should remain in the cloud. The idea is that, once the TTL expires, the data is permanently removed since it is no longer required or relevant and the same should apply with traceability data. The question remains how much of the dependency information should also be removed, and what impact would this have on the overall traceability of the graph. This approach may prove challenging, but can provide a satisfactory level of performance time, and with good scalability.

Visualization of Traceability Data - The cProv model does not define how traceability data should be visually presented. While this was out of the scope for the model, it is also one of the most important factors of usability in the industry. The simpler and more intuitive the representation the better the adaptability would be. The current representation of traceability data is graph-based, which may not necessarily be the best choice for the industry, as it difficult to digest a large graph, requiring a great deal of effort in understanding. A desired approach would be to provide a summarized visual representation of the information as digestives.

Secure Provenance Data - The current framework generates and stores the traceability data in structured text (XML). For our service requirement, this solution is satisfactory since no user has direct access to the Traceability store. However, if in a situation the store was to be compromised, or was to fall into the wrong hands, traceability data would be accessible and could potentially be misused. In the future, traceability data needs to be managed and stored securely.

A Combination of Traceability and Service-level Data - While the language primarily focuses on using the provenance data for validating the rules, it does not consider the service level data. A combination of both types of data may enrich the expressive capabilities of the language further, but may introduce additional complexities, and performance issues, since both types of data are need to be handled simultaneously. However, the current control mechanism is based on reactive enforcement, and it may be possible to introduce proactive enforcement with the additional service data. This would be extremely useful for the business if they can predict and control current and future events, but the success would be highly dependent on the quality and accuracy of such controls.

Integration with Existing Framework - Our framework is currently designed to run on its own to provide traceability and policy enforcement features. However, this requires developers to learn and adopt a new framework, which can be additional work load. Having a partial or full integration with a widely used framework such as Spring would greatly help for its adoption commercially.

Smart Cloud Devices - The concept of smart cloud devices to form mobile cloud services is still in its infancy. However, with the mobile devices increasing in power, storage and computational capabilities, we believe that future smart cloud device-based services are likely to become more homogeneous. This is where a service may be fragmented between multiple smart devices, where they may perform certain tasks either having a permanent or non-permanent connection (local storage and processing takes place) to the Internet, contributing towards overall service goals. Our framework can be extended to empower such service scenarios by providing the traceability capability and policy-based enforcement.

8.3 Concluding Remarks

This thesis has shown how provenance-based data traceability solution can be used for cloud services in addressing access control to sharing, storage and processing of data. It also offers intangible benefits of improving trust of the service provider and an increased level of transparency of the service in the cloud environment. We believe this solution can bring an added value proposition to the service provider and can play an important role in attracting and retaining of customers.

Data traceability-based technology has an important role to play in the cloud environment. The framework presented in this thesis is a stepping stone for commercialization of provenance-powered cloud services. However, greater industrial involvements are required to champion such frameworks to gain mass market adoption of traceability-based solutions. To conclude, data traceability is here to stay and it is only a matter of time before we see proliferation of provenance-based services on the market.

Appendix A

Source Code and Policies Validation

This chapter provides source codes for the traceability model and policy language and APIs used for the framework. It presents various Schemas and Stylesheets used for defining the structure and syntax for the traceability model and policy language and various mappings to the XACML standard. The chapter also provides policy validations for the ConfidenShare service.

A.1 cProv Model XML Schema

```
<?xml version="1.0" encoding="utf-8"?>
<!--Author: Mufy -->
<xs:schema targetNamespace="http://labs.orange.com/uk/cprov#"
        xmlns:prov="http://www.w3.org/ns/prov#" xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:cprov="http://labs.orange.com/uk/cprov#" elementFormDefault="qualified"
        attributeFormDefault="unqualified" xmlns:opmx="http://openprovenance.org/model/opmx#"
        xmlns:cprovd="http://labs.orange.com/uk/cprovd#">
        <xs:import namespace="http://www.w3.org/ns/prov#" schemaLocation="prov-07012013.xsd"/>
        <xs:element name="traceabilityDocument" type="cprov:TraceabilityDocument"> </xs:element>
        <xs:group name="instance">
                <xs:sequence>
                        <xs:choice maxOccurs="unbounded" minOccurs="0">
                                <xs:element ref="cprov:cProcess"/>
                                <xs:element ref="cprov:cResource"/>
                                <xs:element ref="cprov:pResource"/>
                                <xs:element ref="cprov:transition"/>
                        </xs:choice>
                </xs:sequence>
        </xs:group>
        <xs:group name="relation">
                <xs:sequence>
                        <xs:choice maxOccurs="unbounded" minOccurs="0">
                                <xs:element ref="cprov:hadOwnership"/>
                                <xs:element ref="cprov:wasExplicitlyCalledBy"/>
                                <xs:element ref="cprov:wasImplicitlyCalledBy"/>
                                <xs:element ref="cprov:wasInitiallyCalledBy"/>
                                <xs:element ref="cprov:wasRecurrentlyCalledBy"/>
                                <xs:element ref="cprov:wasReferenceOf"/>
                                <xs:element ref="cprov:wasRepresentationOf"/>
                                <xs:element ref="cprov:wasVirtualizedBy"/>
                                <xs:element ref="cprov:hadTransitionState_a"/>
                                <xs:element ref="cprov:hadTransitionState_b"/>
```

```
33                              <xs:element ref="cprov:hadTransitionState_c"/>
34                      </xs:choice>
35              </xs:sequence>
36
37      </xs:group>
38      <xs:complexType name="TraceabilityDocument" mixed="false">
39              <xs:complexContent>
40                      <xs:extension base="prov:Document">
41                              <xs:sequence>
42                                      <xs:group maxOccurs="unbounded" minOccurs="0" ref="cprov:instance"/>
43                                      <xs:group maxOccurs="unbounded" minOccurs="0" ref="cprov:relation"/>
44                              </xs:sequence>
45                      </xs:extension>
46              </xs:complexContent>
47      </xs:complexType>
48      <xs:element name="resource" type="cprov:Resource"/>
49      <xs:complexType name="Resource">
50              <xs:complexContent>
51                      <xs:extension base="prov:Entity">
52                              <xs:sequence>
53                                      <xs:element name="resType" type="cprov:resType" maxOccurs="1" minOccurs="0"/>
54                                      <xs:element name="userTrustDegree" type="xs:float" maxOccurs="1" minOccurs="0"/>
55                                      <xs:element name="restrictionType" type="cprov:restriction" minOccurs="0"
56                                              maxOccurs="2"/>
57                                      <xs:element name="des" type="xs:string" minOccurs="0"/>
58                              </xs:sequence>
59                      </xs:extension>
60              </xs:complexContent>
61      </xs:complexType>
62      <xs:element name="pResource" type="cprov:pResource"/>
63      <xs:complexType name="pResource">
64              <xs:complexContent>
65                      <xs:extension base="cprov:Resource">
66                              <xs:sequence>
67                                      <xs:element name="originIP" type="xs:string" minOccurs="0" maxOccurs="1"/>
68                                      <xs:element name="MAC" type="xs:string" minOccurs="0" maxOccurs="1"/>
69                                      <xs:element name="host" type="cprov:host"/>
70                              </xs:sequence>
71                      </xs:extension>
72              </xs:complexContent>
73      </xs:complexType>
74      <xs:element name="cResource" type="cprov:cResource"/>
75      <xs:complexType name="cResource">
76              <xs:complexContent>
77                      <xs:extension base="cprov:Resource">
78                              <xs:sequence>
79                                      <xs:element name="userCloudRef" type="xs:string"/>
80                                      <xs:element name="vResourceRef" type="xs:string"/>
81                                      <xs:element name="pResourceRef" type="xs:string"/>
82                                      <xs:element name="isReplicable" type="xs:boolean" maxOccurs="1" minOccurs="0"/>
83                                      <xs:element name="TTL" type="xs:dateTime" maxOccurs="1" minOccurs="0"/>
84                              </xs:sequence>
85                      </xs:extension>
86              </xs:complexContent>
87      </xs:complexType>
88      <xs:element name="cProcess" type="cprov:cProcess"/>
89      <xs:complexType name="cProcess">
90              <xs:complexContent>
91                      <xs:extension base="prov:Activity">
92                              <xs:sequence>
93                                      <xs:element name="userCloudRef" type="xs:string" minOccurs="1"/>
94                                      <xs:element name="vProcessRef" type="xs:string" minOccurs="1"/>
95                                      <xs:element name="pProcessRef" type="xs:string" minOccurs="1"/>
96                                      <xs:element name="des" type="xs:string" minOccurs="0"/>
97                              </xs:sequence>
98                      </xs:extension>
99              </xs:complexContent>
100     </xs:complexType>
101     <xs:element name="transition" type="cprov:Transition"/>
102     <xs:complexType name="Transition">
103             <xs:complexContent>
104                     <xs:extension base="prov:Entity">
105                             <xs:sequence>
106                                     <!-- <xs:element name="state" type="xs:QName" maxOccurs="1" minOccurs="1">
107                                             </xs:element> -->
108                                     <xs:element name="state" type="xs:QName"/>
109                                     <xs:element name="latitude" type="xs:string" maxOccurs="1" minOccurs="0"/>
110                                     <xs:element name="longitude" type="xs:string" maxOccurs="1" minOccurs="0"/>
111                                     <xs:element name="country" type="xs:QName" maxOccurs="1" minOccurs="0"/>
112                                     <xs:element name="region" type="xs:QName" maxOccurs="1" minOccurs="0"/>
```

```
                                <xs:element name="event" type="xs:QName"/>
                        </xs:sequence>
                </xs:extension>
        </xs:complexContent>
</xs:complexType>
<xs:element name="wasExplicitlyCalledBy" type="cprov:wasExplicitlyCalledBy"/>
<xs:complexType name="wasExplicitlyCalledBy">
        <xs:complexContent>
                <xs:extension base="prov:Communication">
                        <xs:sequence>
                                <xs:element name="explicitType" type="xs:QName" maxOccurs="1" minOccurs="0"/>
                                <xs:element name="callComm" type="cprov:comm" maxOccurs="1" minOccurs="0"/>
                                <xs:element name="callMedium" type="cprov:host" maxOccurs="1" minOccurs="0"/>
                                <xs:element name="callNetwork" type="cprov:network" maxOccurs="1" minOccurs="0"
                                />
                        </xs:sequence>
                </xs:extension>
        </xs:complexContent>
</xs:complexType>
<xs:element name="wasImplicitlyCalledBy" type="cprov:wasImplicitlyCalledBy"/>
<xs:complexType name="wasImplicitlyCalledBy">
        <xs:complexContent>
                <xs:extension base="prov:Communication">
                        <xs:sequence>
                                <xs:element name="implicitType" type="xs:QName" minOccurs="0"/>
                                <xs:element name="callComm" type="cprov:comm" maxOccurs="1" minOccurs="0"/>
                                <xs:element name="callMedium" type="cprov:host" maxOccurs="1" minOccurs="0"/>
                                <xs:element name="callNetwork" type="cprov:network" maxOccurs="1" minOccurs="0"
                                />
                        </xs:sequence>
                </xs:extension>
        </xs:complexContent>
</xs:complexType>
<xs:element name="wasRecurrentlyCalledBy" type="cprov:wasRecurrentlyCalledBy"/>
<xs:complexType name="wasRecurrentlyCalledBy">
        <xs:complexContent>
                <xs:extension base="prov:Communication">
                        <xs:sequence>
                                <xs:element name="recurrType" type="xs:QName"/>
                                <xs:element name="timeOut" type="xs:dateTime" maxOccurs="1" minOccurs="0"/>
                                <xs:element name="callComm" type="xs:QName" maxOccurs="1" minOccurs="0"/>
                                <xs:element name="callMedium" type="cprov:host" maxOccurs="1" minOccurs="0"/>
                                <xs:element name="callNetwork" type="cprov:network" maxOccurs="1" minOccurs="0"
                                />
                        </xs:sequence>
                </xs:extension>
        </xs:complexContent>
</xs:complexType>
<xs:element name="wasRepresentationOf" type="cprov:wasRepresentationOf"> </xs:element>
<xs:complexType name="wasRepresentationOf">
        <xs:complexContent>
                <xs:extension base="prov:Derivation">
                        <xs:sequence>
                                <xs:element name="method" type="xs:QName" minOccurs="0"/>
                        </xs:sequence>
                </xs:extension>
        </xs:complexContent>
</xs:complexType>
<xs:element name="wasReferenceOf" type="cprov:wasReferenceOf"/>
<xs:complexType name="wasReferenceOf">
        <xs:complexContent>
                <xs:extension base="prov:Derivation">
                        <xs:sequence>
                                <xs:element name="refType" type="xs:QName"/>
                                <xs:element name="method" type="xs:QName" minOccurs="0"/>
                                <xs:element name="restrictionType" type="cprov:restriction"/>
                        </xs:sequence>
                </xs:extension>
        </xs:complexContent>
</xs:complexType>
<xs:element name="wasInitiallyCalledBy" type="cprov:wasInitiallyCalledBy"/>
<xs:complexType name="wasInitiallyCalledBy">
        <xs:complexContent>
                <xs:extension base="prov:Association">
                        <xs:sequence>
                                <xs:element name="purpose" type="xs:QName" maxOccurs="1" minOccurs="0"/>
                                <xs:element name="accessMedium" type="cprov:host" maxOccurs="1" minOccurs="0"/>
                                <xs:element name="accessNetwork" type="cprov:network" maxOccurs="1"
                                        minOccurs="0"/>
                        </xs:sequence>
```

```
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:element name="wasVirtualizedBy" type="cprov:wasVirtualizedBy"/>
        <xs:complexType name="wasVirtualizedBy">
            <xs:complexContent>
                <xs:extension base="prov:Generation">
                    <xs:sequence>
                        <xs:element name="purpose" type="xs:QName" minOccurs="0"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>

        <!-- TODO - change the type name (spelled wrong ) -->
        <xs:element name="hadOwnership" type="cprov:hadOwnership"/>
        <xs:complexType name="hadOwnership">
            <xs:complexContent>
                <xs:extension base="prov:Attribution">
                    <xs:sequence>
                        <xs:element name="ownershipType" type="cprov:ownership"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:element name="hadTransitionState_a" type="cprov:hadTransitionState_a"/>
        <xs:element name="hadTransitionState_b" type="cprov:hadTransitionState_b"/>
        <xs:element name="hadTransitionState_c" type="cprov:hadTransitionState_c"/>
        <xs:complexType name="hadTransitionState_a">
            <xs:complexContent>
                <xs:extension base="prov:Derivation">
                    <xs:sequence>
                        <xs:element name="method" type="xs:QName" minOccurs="0"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="hadTransitionState_b">
            <xs:complexContent>
                <xs:extension base="prov:Generation">
                    <xs:sequence>
                        <xs:element name="method" type="xs:QName" minOccurs="0"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="hadTransitionState_c">
            <xs:complexContent>
                <xs:extension base="prov:Attribution">
                    <xs:sequence>
                        <xs:element name="method" type="xs:QName" minOccurs="0"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:attribute name="id" type="xs:QName"/>
        <xs:simpleType name="ownership">
            <xs:restriction base="xs:QName">
                <xs:enumeration value="cprovd:originator"/>
                <xs:enumeration value="cprovd:possession"/>
                <xs:enumeration value="cprovd:affiliation"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="restriction">
            <xs:restriction base="xs:QName">
                <xs:enumeration value="cprovd:confidential"/>
                <xs:enumeration value="cprovd:restricted"/>
                <xs:enumeration value="cprovd:general"/>
                <xs:enumeration value="cprovd:non-modifiable"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="resType">
            <xs:restriction base="xs:QName">
                <xs:enumeration value="cprovd:data"/>
                <xs:enumeration value="cprovd:file"/>
                <xs:enumeration value="cprovd:text"/>
                <xs:enumeration value="cprovd:audio"/>
                <xs:enumeration value="cprovd:video"/>
                <xs:enumeration value="cprovd:image"/>
                <xs:enumeration value="cprovd:binary"/>
```

```
273             </xs:restriction>
274     </xs:simpleType>
275     <xs:simpleType name="host">
276             <xs:restriction base="xs:QName">
277                     <xs:enumeration value="cprovd:pc"/>
278                     <xs:enumeration value="cprovd:mobile"/>
279                     <xs:enumeration value="cprovd:laptop"/>
280                     <xs:enumeration value="cprovd:tablet"/>
281             </xs:restriction>
282     </xs:simpleType>
283     <xs:simpleType name="state">
284             <xs:restriction base="xs:QName">
285                     <xs:enumeration value="cprovd:origin"/>
286                     <xs:enumeration value="cprovd:source"/>
287                     <xs:enumeration value="cprovd:intermediary"/>
288                     <xs:enumeration value="cprovd:destination"/>
289             </xs:restriction>
290     </xs:simpleType>
291     <xs:simpleType name="comm">
292             <xs:restriction base="xs:QName">
293                     <xs:enumeration value="cprovd:synchronized"/>
294                     <xs:enumeration value="cprovd:asynchronized"/>
295             </xs:restriction>
296     </xs:simpleType>
297     <xs:simpleType name="network">
298             <xs:restriction base="xs:QName">
299                     <xs:enumeration value="cprovd:thirdGeneration"/>
300                     <xs:enumeration value="cprovd:fourthGeneration"/>
301                     <xs:enumeration value="cprovd:wifi"/>
302                     <xs:enumeration value="cprovd:modem"/>
303                     <xs:enumeration value="cprovd:ISDN"/>
304                     <xs:enumeration value="cprovd:DSL"/>
305                     <xs:enumeration value="cprovd:satelite"/>
306             </xs:restriction>
307     </xs:simpleType>
308     <xs:simpleType name="purpose">
309             <xs:restriction base="xs:QName">
310                     <xs:enumeration value="cprovd:registration"/>
311                     <xs:enumeration value="cprovd:administration"/>
312                     <xs:enumeration value="cprovd:verification"/>
313                     <xs:enumeration value="cprovd:validation"/>
314                     <xs:enumeration value="cprovd:operation"/>
315             </xs:restriction>
316     </xs:simpleType>
317 </xs:schema>
```

A.2 cProvl Policy Language XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Author: Mufy -->
<xs:schema targetNamespace="http://labs.orange.com/uk/cprovl#"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
    attributeFormDefault="unqualified" xmlns:cprov="http://labs.orange.com/uk/cprov#"
    xmlns:cprovl="http://labs.orange.com/uk/cprovl#" xmlns:prov="http://www.w3.org/ns/prov#"
    xmlns:cprovd="http://labs.orange.com/uk/cprovd#" xmlns:r="http://labs.orange.com/uk/r#">
    <xs:import namespace="http://labs.orange.com/uk/cprov#"
        schemaLocation="../cProv-inherited/cProv-v1.3.xsd"/>
    <xs:import namespace="http://www.w3.org/ns/prov#"
        schemaLocation="../cProv-inherited/prov-07012013.xsd"/>
    <xs:element name="POLICY" type="cprovl:Policy"> </xs:element>
    <xs:complexType name="Policy">
        <xs:sequence>
            <xs:element name="entity" type="cprovl:u-Entity"/>
            <xs:element ref="cprovl:ENV"/>
            <xs:element ref="cprovl:RULE" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="RULE" type="cprovl:Rule"> </xs:element>
    <xs:element name="INHERIT" type="cprovl:Inherit"> </xs:element>
    <xs:element name="REFERENCE" type="cprovl:Override"> </xs:element>
    <xs:element name="DEF" type="cprovl:Def"> </xs:element>
    <xs:element name="IF" type="cprovl:targets"> </xs:element>
    <xs:element name="THEN" type="cprovl:execution"> </xs:element>
    <xs:complexType name="Rule">
        <xs:sequence>
            <xs:element name="entity" type="cprovl:u-Entity"/>
            <xs:choice>
                <xs:element ref="cprovl:INHERIT" maxOccurs="1" minOccurs="0"/>
                <xs:element ref="cprovl:REFERENCE"/>
            </xs:choice>
            <xs:element ref="cprovl:DEF" maxOccurs="1" minOccurs="1"/>
            <xs:element ref="cprovl:IF" maxOccurs="1" minOccurs="0"/>
            <xs:element ref="cprovl:SUCH-THAT" maxOccurs="1" minOccurs="0"/>
            <xs:element ref="cprovl:THEN" maxOccurs="1" minOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="targets">
        <xs:sequence>
            <xs:choice maxOccurs="unbounded">
                <xs:element form="qualified" name="target" type="cprovl:identifier"
                    maxOccurs="unbounded" minOccurs="0"/>
            </xs:choice>
        </xs:sequence>
        <xs:attribute use="required" ref="cprovl:quantifier"/>
    </xs:complexType>
    <xs:complexType name="identifier">
        <xs:choice>
            <xs:element name="ID" maxOccurs="unbounded" minOccurs="1">
                <xs:complexType>
                    <xs:simpleContent>
                        <xs:extension base="xs:QName">
                            <xs:attribute use="optional" ref="cprovl:var_type"> </xs:attribute>
                            <xs:attribute use="optional" ref="cprovl:var_identifier"/>
                            <xs:attribute ref="cprovl:category_type"/>
                        </xs:extension>
                    </xs:simpleContent>
                </xs:complexType>
            </xs:element>
        </xs:choice>
        <xs:attribute default="cprovd:And" ref="cprovl:l_operator"/>
    </xs:complexType>
    <xs:complexType name="condition">
        <xs:sequence>
            <xs:element ref="cprovl:statements" maxOccurs="unbounded" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="SUCH-THAT" type="cprovl:condition"> </xs:element>
    <xs:complexType name="Inherit">
        <xs:sequence maxOccurs="unbounded">
            <xs:element form="qualified" name="var" type="cprovl:var"/>
            <xs:element name="entity" type="cprovl:i-Entity"/>
        </xs:sequence>
    </xs:complexType>
```

```
76      <xs:complexType name="Override">
77          <xs:sequence>
78              <xs:element form="qualified" name="var" type="cprovl:var"/>
79              <xs:element name="entity" type="cprovl:o-Entity"/>
80          </xs:sequence>
81      </xs:complexType>
82      <xs:complexType name="Before">
83          <xs:sequence maxOccurs="unbounded">
84              <xs:element form="qualified" name="var" type="cprovl:var"/>
85              <xs:element ref="prov:entity"/>
86          </xs:sequence>
87      </xs:complexType>
88      <xs:complexType name="After">
89          <xs:sequence maxOccurs="unbounded">
90              <xs:element form="qualified" name="var" type="cprovl:var"/>
91              <xs:element ref="prov:entity"/>
92          </xs:sequence>
93      </xs:complexType>
94      <xs:complexType name="Def">
95          <xs:sequence>
96              <xs:element name="entity" type="cprovl:d-Entity"/>
97          </xs:sequence>
98      </xs:complexType>
99      <xs:element name="statements" type="cprovl:Statements"/>
100     <xs:complexType name="Statements">
101         <xs:sequence maxOccurs="unbounded">
102             <xs:choice>
103                 <xs:element name="statement" type="cprovl:Statement"/>
104                 <xs:element ref="cprovl:grouping" minOccurs="0"/>
105                 <xs:element ref="cprovl:operator" minOccurs="0"/>
106             </xs:choice>
107         </xs:sequence>
108     </xs:complexType>
109     <xs:complexType name="execution">
110         <xs:sequence>
111             <xs:element form="qualified" name="var" type="cprovl:var" minOccurs="0"
112                 maxOccurs="unbounded"/>
113             <xs:element maxOccurs="2" name="entity" type="cprovl:e-Entity"/>
114             <xs:element maxOccurs="unbounded" ref="cprov:hadOwnership"/>
115         </xs:sequence>
116     </xs:complexType>
117     <xs:element name="executionStatement" type="cprovl:ExecutionStatement"/>
118     <xs:complexType name="ExecutionStatement">
119         <xs:sequence>
120             <xs:element form="qualified" name="var" type="cprovl:var"/>
121             <xs:element name="statement" type="cprovl:Statement"/>
122         </xs:sequence>
123     </xs:complexType>
124     <xs:complexType name="Statement">
125         <xs:complexContent>
126             <xs:extension base="cprov:TraceabilityDocument">
127                 <xs:choice>
128                     <xs:element name="var" type="cprovl:var" minOccurs="0" maxOccurs="unbounded"/>
129                 </xs:choice>
130                 <xs:attribute ref="cprovl:negate"/>
131                 <xs:attribute ref="cprovl:category_type"/>
132             </xs:extension>
133         </xs:complexContent>
134     </xs:complexType>
135     <xs:element name="grouping" type="cprovl:Grouping"> </xs:element>
136     <xs:complexType name="Grouping">
137         <xs:sequence maxOccurs="unbounded">
138             <xs:choice>
139                 <xs:element ref="cprovl:grouping"/>
140                 <xs:element name="statement" type="cprovl:Statement"/>
141                 <xs:element ref="cprovl:operator" minOccurs="0"/>
142             </xs:choice>
143         </xs:sequence>
144         <xs:attribute ref="cprovl:l_operator"/>
145     </xs:complexType>
146     <xs:element name="operator" type="cprovl:ConOperator"/>
147     <xs:complexType name="ConOperator">
148         <xs:sequence>
149             <xs:element name="statement" type="cprovl:Statement" maxOccurs="2" minOccurs="0"/>
150         </xs:sequence>
151         <xs:attribute ref="cprovl:c_operator"/>
152     </xs:complexType>
153     <xs:element name="PolicyRequest" type="cprovl:policyRequest"> </xs:element>
154     <xs:complexType name="policyRequest">
155         <xs:sequence>
```

```
156             <xs:element name="Agent" maxOccurs="unbounded" minOccurs="1">
157                 <xs:complexType>
158                     <xs:complexContent>
159                         <xs:extension base="prov:Agent">
160                             <xs:attribute form="unqualified" name="isRef" type="xs:boolean"/>
161                         </xs:extension>
162                     </xs:complexContent>
163                 </xs:complexType>
164             </xs:element>
165             <xs:element name="Entity" nillable="false" type="cprovl:r-Entity" maxOccurs="unbounded"/>
166             <xs:element maxOccurs="4" minOccurs="1" ref="cprov:hadOwnership"/>
167         </xs:sequence>
168     </xs:complexType>
169     <xs:simpleType name="categoryType">
170         <xs:restriction base="xs:QName">
171             <xs:enumeration value="cprovd:Service"/>
172             <xs:enumeration value="cprovd:Subject"/>
173             <xs:enumeration value="cprovd:Resource"/>
174             <xs:enumeration value="cprovd:Action"/>
175             <xs:enumeration value="cprovd:Environment"/>
176         </xs:restriction>
177     </xs:simpleType>
178     <xs:simpleType name="categoryValue">
179         <xs:restriction base="xs:string"/>
180     </xs:simpleType>
181     <xs:element name="policyResponse" type="cprovl:policyResponse"/>
182     <xs:complexType name="policyResponse">
183         <xs:sequence>
184             <xs:element name="Agent">
185                 <xs:complexType>
186                     <xs:complexContent>
187                         <xs:extension base="prov:Agent">
188                             <xs:attribute form="unqualified" name="isRef" type="xs:boolean"/>
189                         </xs:extension>
190                     </xs:complexContent>
191                 </xs:complexType>
192             </xs:element>
193             <xs:element name="Entity" type="cprovl:rs-Entity"/>
194             <xs:element maxOccurs="unbounded" ref="cprov:cResource"/>
195             <xs:element maxOccurs="unbounded" ref="cprov:cProcess"/>
196             <xs:element ref="cprov:transition"/>
197             <xs:element ref="cprov:hadOwnership"/>
198             <xs:element ref="cprov:wasReferenceOf"/>
199             <xs:element ref="cprov:wasImplicitlyCalledBy" maxOccurs="unbounded"/>
200             <xs:element ref="cprov:wasVirtualizedBy" maxOccurs="unbounded"/>
201             <xs:element ref="cprov:wasInitiallyCalledBy"/>
202             <xs:element ref="prov:used"/>
203             <xs:element ref="cprov:hadTransitionState_c"/>
204         </xs:sequence>
205     </xs:complexType>
206     <xs:simpleType name="logicalOperator">
207         <xs:restriction base="xs:QName">
208             <xs:enumeration value="cprovd:And"/>
209             <xs:enumeration value="cprovd:Or"/>
210             <xs:enumeration value="cprovd:Not"/>
211             <xs:enumeration value="cprovd:Implication"/>
212         </xs:restriction>
213     </xs:simpleType>
214     <xs:complexType name="targetID">
215         <xs:attribute name="newAttribute"/>
216     </xs:complexType>
217     <xs:complexType name="var">
218         <xs:attribute use="required" ref="cprovl:var_type"> </xs:attribute>
219         <xs:attribute use="required" ref="cprovl:var_identifier"/>
220     </xs:complexType>
221     <xs:simpleType name="varType">
222         <xs:restriction base="xs:QName">
223             <xs:enumeration value="cprovd:new"/>
224             <xs:enumeration value="cprovd:s-ref"/>
225             <xs:enumeration value="cprovd:d-ref"/>
226         </xs:restriction>
227     </xs:simpleType>
228     <xs:simpleType name="Identifier">
229         <xs:restriction base="xs:QName"/>
230     </xs:simpleType>
231     <xs:simpleType name="range">
232         <xs:restriction base="xs:QName">
233             <xs:enumeration value="cprovd:event"/>
234             <xs:enumeration value="cprovd:All"/>
235             <xs:enumeration value="cprovd:Node"/>
```

```
236             <xs:enumeration value="cprovd:Edge"/>
237             <xs:enumeration value="cprovd:Entity"/>
238             <xs:enumeration value="cprovd:Activity"/>
239             <xs:enumeration value="cprovd:Agent"/>
240             <xs:enumeration value="cprovd:Wgb"/>
241             <xs:enumeration value="cprovd:WdF"/>
242             <xs:enumeration value="cprovd:WaW"/>
243             <xs:enumeration value="cprovd:WaT"/>
244             <xs:enumeration value="cprovd:WiB"/>
245             <xs:enumeration value="cprovd:Used"/>
246         </xs:restriction>
247     </xs:simpleType>
248     <xs:complexType name="u-Entity">
249         <xs:complexContent>
250             <xs:extension base="prov:Entity">
251                 <xs:choice>
252                     <xs:element name="description" type="cprovl:description"/>
253                 </xs:choice>
254             </xs:extension>
255         </xs:complexContent>
256     </xs:complexType>
257     <xs:complexType name="d-Entity">
258         <xs:complexContent>
259             <xs:extension base="prov:Entity">
260                 <xs:choice>
261                     <xs:element name="range" type="cprovl:range"/>
262                 </xs:choice>
263             </xs:extension>
264         </xs:complexContent>
265     </xs:complexType>
266     <xs:complexType name="e-Entity">
267         <xs:complexContent>
268             <xs:extension base="prov:Entity">
269                 <xs:choice>
270                     <xs:sequence>
271                         <xs:element name="actionType" type="cprovl:actionType"/>
272                         <xs:element name="resourceId" type="cprovl:resourceID"/>
273                         <xs:sequence>
274                             <xs:element name="obligationType" type="cprovl:obligationType"
275                                 minOccurs="0"/>
276                             <xs:element name="obligationAction" type="cprovl:obligationAction"
277                                 minOccurs="0"/>
278                         </xs:sequence>
279                     </xs:sequence>
280                     <xs:element name="description" type="cprovl:description"/>
281                 </xs:choice>
282             </xs:extension>
283         </xs:complexContent>
284     </xs:complexType>
285     <xs:complexType name="i-Entity">
286         <xs:complexContent>
287             <xs:extension base="prov:Entity">
288                 <xs:choice>
289                     <xs:element name="rule" type="xs:QName"/>
290                     <xs:element name="part" type="xs:QName"/>
291                 </xs:choice>
292             </xs:extension>
293         </xs:complexContent>
294     </xs:complexType>
295     <xs:complexType name="o-Entity">
296         <xs:complexContent>
297             <xs:extension base="prov:Entity">
298                 <xs:choice>
299                     <xs:element name="rule" type="xs:QName"/>
300                     <xs:element name="order" type="cprovl:order"/>
301                 </xs:choice>
302             </xs:extension>
303         </xs:complexContent>
304     </xs:complexType>
305     <xs:element name="ENV" type="cprovl:ENV"/>
306     <xs:complexType name="ENV">
307         <xs:sequence>
308             <xs:element name="entity" type="cprovl:u-Entity"/>
309         </xs:sequence>
310     </xs:complexType>
311     <xs:simpleType name="actionType">
312         <xs:restriction base="xs:QName">
313             <xs:enumeration value="cprovd:Permit"/>
314             <xs:enumeration value="cprovd:Deny"/>
315             <xs:enumeration value="cprovd:Indeterminate"/>
```

```
316             <xs:enumeration value="cprovd:Not-applicable"/>
317             <xs:enumeration value="cprovd:Obligation"/>
318         </xs:restriction>
319     </xs:simpleType>
320     <xs:simpleType name="resourceID">
321         <xs:restriction base="xs:QName"/>
322     </xs:simpleType>
323     <xs:simpleType name="obligationType">
324         <xs:restriction base="xs:QName">
325             <xs:enumeration value="cprovd:Alert"/>
326             <xs:enumeration value="cprovd:Notify"/>
327         </xs:restriction>
328     </xs:simpleType>
329     <xs:simpleType name="obligationAction">
330         <xs:restriction base="xs:QName">
331             <xs:enumeration value="cprovd:Record"/>
332             <xs:enumeration value="cprovd:Delete"/>
333             <xs:enumeration value="cprovd:Copy"/>
334             <xs:enumeration value="cprovd:Read-only"/>
335             <xs:enumeration value="cprovd:Read-write"/>
336             <xs:enumeration value="cprovd:check-user-history"/>
337             <xs:enumeration value="cprovd:Non-share"/>
338             <xs:enumeration value="cprovd:Block-user"/>
339             <xs:enumeration value="cprovd:Remove-user"/>
340         </xs:restriction>
341     </xs:simpleType>
342     <xs:simpleType name="obligationSet">
343         <xs:restriction base="xs:QName">
344             <xs:enumeration value="cprovd:Pre"/>
345             <xs:enumeration value="cprovd:Post"/>
346         </xs:restriction>
347     </xs:simpleType>
348     <xs:simpleType name="msg">
349         <xs:restriction base="xs:string"/>
350     </xs:simpleType>
351     <xs:simpleType name="description">
352         <xs:restriction base="xs:string"/>
353     </xs:simpleType>
354     <xs:simpleType name="conditonalOperator">
355         <xs:restriction base="xs:QName">
356             <xs:enumeration value="cprovd:Equal-to"/>
357             <xs:enumeration value="cprovd:Greater-than"/>
358             <xs:enumeration value="cprovd:Less-than"/>
359             <xs:enumeration value="cprovd:Not-equal-to"/>
360             <xs:enumeration value="cprovd:Greater-than-or-equal-to"/>
361             <xs:enumeration value="cprovd:Less-than-or-equal-to"/>
362         </xs:restriction>
363     </xs:simpleType>
364     <xs:simpleType name="targetQuantifier">
365         <xs:restriction base="xs:QName">
366             <xs:enumeration value="cprovd:There-exist"/>
367             <xs:enumeration value="cprovd:For-all"/>
368         </xs:restriction>
369     </xs:simpleType>
370     <xs:attribute name="var_type" type="cprovl:varType"/>
371     <xs:attribute name="var_identifier" type="cprovl:Identifier"/>
372     <xs:attribute name="category_type" type="cprovl:categoryType"/>
373     <xs:attribute name="l_operator" type="cprovl:logicalOperator"/>
374     <xs:attribute name="c_operator" type="cprovl:conditonalOperator"/>
375     <xs:attribute name="quantifier" type="cprovl:targetQuantifier"/>
376     <xs:attribute default="false" name="negate" type="xs:boolean"/>
377     <xs:simpleType name="part">
378         <xs:restriction base="xs:QName">
379             <xs:enumeration value="cprovd:Condition"/>
380             <xs:enumeration value="cprovd:Execution"/>
381         </xs:restriction>
382     </xs:simpleType>
383     <xs:simpleType name="order">
384         <xs:restriction base="xs:QName">
385             <xs:enumeration value="cprovd:Before"/>
386             <xs:enumeration value="cprovd:After"/>
387         </xs:restriction>
388     </xs:simpleType>
389
390     <xs:simpleType name="requestField">
391         <xs:restriction base="xs:QName">
392             <xs:enumeration value="cprovd:Service"/>
393             <xs:enumeration value="cprovd:Subject"/>
394             <xs:enumeration value="cprovd:Resource"/>
395             <xs:enumeration value="cprovd:Action"/>
```

```
396             <xs:enumeration value="cprovd:Environment"/>
397         </xs:restriction>
398     </xs:simpleType>
399     <xs:simpleType name="fieldValue">
400         <xs:restriction base="xs:string"/>
401     </xs:simpleType>
402 
403     <xs:complexType name="r-Entity">
404         <xs:complexContent>
405             <xs:extension base="prov:Entity">
406                 <xs:sequence>
407                     <xs:element name="categoryType" type="cprovl:requestField"/>
408                     <xs:element name="categoryValue">
409                         <xs:complexType>
410                             <xs:simpleContent>
411                                 <xs:extension base="cprovl:fieldValue">
412                                     <xs:attribute name="isRef" type="xs:boolean"/>
413                                 </xs:extension>
414                             </xs:simpleContent>
415                         </xs:complexType>
416                     </xs:element>
417                     <xs:element name="description" type="cprovl:description"/>
418                 </xs:sequence>
419             </xs:extension>
420         </xs:complexContent>
421     </xs:complexType>
422     <xs:complexType name="rs-Entity">
423         <xs:complexContent>
424             <xs:extension base="prov:Entity">
425                 <xs:sequence>
426                     <xs:element name="service" type="xs:QName"/>
427                     <xs:element name="decision" type="cprovl:actionType"/>
428                     <xs:element name="description" type="xs:string"/>
429                 </xs:sequence>
430             </xs:extension>
431         </xs:complexContent>
432     </xs:complexType>
433 </xs:schema>
```

A.3 cProv to XACML policy XSLT Transformer

```
<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:cprov="http://labs.orange.com/uk/cprov#" xmlns:cprovl="http://labs.orange.com/uk/cprovl#"
 xmlns:prov="http://www.w3.org/ns/prov#" xmlns:cprovd="http://labs.orange.com/uk/cprovd#"
 xmlns:r="http://labs.orange.com/uk/r#" xmlns:cu="http://www.w3.org/1999/xhtml/datatypes/"
 xmlns:ex="http://labs.orange.com/uk/ex#" xmlns:xd="http://www.oxygenxml.com/ns/doc/xsl"
 exclude-result-prefixes="xs xd" version="2.0">
   <xd:doc scope="stylesheet">
    <xd:desc>
     <xd:p>
     <xd:b>Updated on:</xd:b>Feb 12, 2013</xd:p>
     <xd:p>
       <xd:b>Author:</xd:b>Mufajjul Ali</xd:p>
     <xd:p />
    </xd:desc>
  </xd:doc>

  <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes" />

  <xsl:template match="/">
    <Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
      ↪    file:/Users/mufy/Dropbox/EngD/year3/schema/cProvl/xacml/xacml-core-v3-schema-wd-17.xsd"
     PolicyId="urn:oasis:names:tc:xacml:3.0:example:SimplePolicy1"
     Version="1.0" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable">

     <xsl:attribute name="PolicyId">
         <xsl:text>urn:oasis:names:tc:xacml:3.0:cprovl:</xsl:text>
         <xsl:value-of select="cprovl:POLICY/cprovl:entity/@prov:id" />
       </xsl:attribute>

     <Description>
       <xsl:value-of select="cprovl:POLICY/cprovl:entity/cprovl:description" />
     </Description>

     <PolicyDefaults>
       <XPathVersion>http://www.w3.org/TR/1999/Rec-xpath-19991116
       </XPathVersion>
     </PolicyDefaults>

     <Target />
       <xsl:for-each select="cprovl:POLICY/cprovl:RULE">
        <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:SimpleRule1" Effect="Permit">
          <xsl:attribute name="RuleId">
               <xsl:text>urn:oasis:names:tc:xacml:3.0:</xsl:text>
               <xsl:value-of select="cprovl:entity/@prov:id " />
             </xsl:attribute>
          <xsl:attribute name="Effect">
               <xsl:variable name="temp" select="cprovl:THEN/cprovl:entity/cprovl:actionType" />
               <xsl:value-of select=" tokenize($temp, ':')[position()=2] " />
             </xsl:attribute>

          <Description>
           <xsl:value-of select="cprovl:entity/cprovl:description" />
          </Description>

          <xsl:if test="count(cprovl:IF)!=0">
           <Target>
             <!-- converting cProvl target to XACML target -->
             <!-- Find all the IDs, and check if they are variables or elements -->
             <xsl:for-each select="cprovl:IF/cprovl:target">
              <xsl:variable name="fieldType" select="  child::cprovl:ID/@cprovl:category_type" />
              <xsl:variable name="varType" select="child::cprovl:ID/@cprovl:var_type" />

              <xsl:if test="$varType='cprovd:new'">
                <AnyOf>
                 <AllOf>
                    <Match MatchId="labs:orange:com:function:ext:xpath-provenance-d-id-match">
                    <AttributeValue DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
                       XPathCategory="urn:com:provenance:cloudprovenance:cprovl:1.0:attribute-category:cprov:newvar">
                      <!-- get the identifier and element -->
                      <xsl:value-of select="child::cprovl:ID" />
```

```
                    </AttributeValue>
                    <AttributeDesignator MustBePresent="false" AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
                      DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression">

                      <!-- Check to see the type of resource -->
                      <xsl:attribute name="Category">
                              <xsl:text>urn:com:provenance:cloudprovenance:cprovl:1.0:attribute-category:cprov:newvar</xsl:text>
                            </xsl:attribute>
                       <xsl:attribute name="AttributeId">
                              <xsl:text>urn:com:provenance:cloudprovenance:cprovl:1.0:attribute-category:cprov:newvar</xsl:text>
                            </xsl:attribute>
                    </AttributeDesignator>
                   </Match>
                  </AllOf>
                 </AnyOf>
                </xsl:if>

                 <xsl:if test=" $varType='cprovd:d-ref' or $varType='cprovd:s-ref'">
                 <!-- Get resouce element which are not referenced as variables -->
                 <AnyOf>
                  <AllOf>
                    <Match
                       MatchId="labs:orange:com:function:ext:xpath-provenance-d-id-match">
                                  <AttributeValue
                                  XPathCategory="urn:com:provenance:cloudprovenance:cprovl:1.0:attribute-category:cprov"
                                  DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
                                   ↪    AttributeId="urn:com:provenance:cloudprovenance:cprovl:1.0:attribute-category:cprov:ref">

                                     <xsl:choose>
                                        <xsl:when test=" $varType='cprovd:d-ref'">

                                           <xsl:attribute name="AttributeId">
                                              <xsl:text>urn:com:provenance:cloudprovenance:cprovl:1.0:attribute-category:cprov:d-ref</xsl:text>
                                           </xsl:attribute>

                                        </xsl:when>
                                        <xsl:otherwise>
                                           <xsl:attribute name="AttributeId">
                                              <xsl:text>urn:com:provenance:cloudprovenance:cprovl:1.0:attribute-category:cprov:s-ref</xsl:text>
                                           </xsl:attribute>
                                        </xsl:otherwise>
                                     </xsl:choose>


                                  <xsl:if test="$fieldType = 'cprovd:Action'">
                                     <xsl:attribute name="XPathCategory">
                                        <xsl:text>urn:oasis:names:tc:xacml:3.0:attribute-category:action</xsl:text>
                                     </xsl:attribute>
                                  </xsl:if>

                                  <xsl:if test="$fieldType = 'cprovd:Service'">
                                     <xsl:attribute name="XPathCategory">
                                        <xsl:text>urn:com:provenance:cloudprovenance:cprovl:1.0:attribute-category:service</xsl:text>
                                     </xsl:attribute>
                                  </xsl:if>
                                  <xsl:if test="$fieldType = 'cprovd:Resource'">
                                     <xsl:attribute name="XPathCategory">
                                        <xsl:text>urn:oasis:names:tc:xacml:3.0:attribute-category:resource</xsl:text>
                                     </xsl:attribute>
                                  </xsl:if>
                                  <xsl:if test="$fieldType = 'cprovd:Subject'">
                                     <xsl:attribute name="XPathCategory">
                                     <xsl:text>urn:oasis:names:tc:xacml:1.0:subject-category:access-subject</xsl:text>
                                  </xsl:attribute>
                                  </xsl:if>

                                  <xsl:if test="not(boolean($fieldType))">
                                     <xsl:attribute name="XPathCategory">
                                        <xsl:text>urn:com:provenance:cloudprovenance:cprovl:1.0:attribute-category:cprov</xsl:text>
                                     </xsl:attribute>
                                  </xsl:if>

                                  <xsl:value-of select="child::cprovl:ID"/>
                                  </AttributeValue>

                     <AttributeDesignator MustBePresent="false"
                       AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
                       DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression">

                       <!-- Check to see the type of resource -->
```

```
154                     <xsl:if test="$fieldType='cprovd:Action'">
155                      <xsl:attribute name="Category">
156                                <xsl:text>urn:oasis:names:tc:xacml:3.0:attribute-category:action</xsl:text>
157                               </xsl:attribute>
158
159                      <xsl:attribute name="AttributeId">
160                                <xsl:text>urn:oasis:names:tc:xacml:3.0:action-id</xsl:text>
161                               </xsl:attribute>
162                     </xsl:if>
163
164                     <xsl:if test="$fieldType='cprovd:Service'">
165                       <xsl:attribute name="Category">
166
                                  ↪    <xsl:text>urn:com:provenance:cloudprovenance:cprovl:1.0:attribute-category:service</xsl:text>
167                               </xsl:attribute>
168
169                      <xsl:attribute name="AttributeId">
170                                <xsl:text>urn:oasis:names:tc:xacml:3.0:service-id</xsl:text>
171                               </xsl:attribute>
172                             </xsl:if>
173
174                     <xsl:if test="$fieldType='cprovd:Resource'">
175                      <xsl:attribute name="Category">
176                                <xsl:text>urn:oasis:names:tc:xacml:3.0:attribute-category:resource</xsl:text>
177                               </xsl:attribute>
178
179                       <xsl:attribute name="AttributeId">
180                                <xsl:text>urn:oasis:names:tc:xacml:3.0:resource-id</xsl:text>
181                               </xsl:attribute>
182                     </xsl:if>
183
184                     <xsl:if test="$fieldType='cprovd:Subject'">
185                      <xsl:attribute name="Category">
186                                <xsl:text>urn:oasis:names:tc:xacml:1.0:subject-category:access-subject</xsl:text>
187                               </xsl:attribute>
188
189                      <xsl:attribute name="AttributeId">
190                                <xsl:text>urn:oasis:names:tc:xacml:3.0:subject-id</xsl:text>
191                               </xsl:attribute>
192                             </xsl:if>
193
194                     <xsl:if test="not(boolean($fieldType))">
195                       <xsl:attribute name="Category">
196                                <xsl:text>urn:com:provenance:cloudprovenance:cprovl:1.0:attribute-category:cprov</xsl:text>
197                               </xsl:attribute>
198
199                      <xsl:attribute name="AttributeId">
200                                <xsl:text>urn:oasis:names:tc:xacml:3.0:cprov-id</xsl:text>
201                               </xsl:attribute>
202                     </xsl:if>
203                    </AttributeDesignator>
204                   </Match>
205                  </AllOf>
206                 </AnyOf>
207             </xsl:if>
208
209             <!-- Handle default IDs -->
210               <xsl:if test="not ($varType)">
211                  <AnyOf>
212                   <AllOf>
213                     <Match MatchId="labs:orange:com:function:ext:xpath-provenance-id-match">
214                      <AttributeValue
215                       XPathCategory="urn:com:provenance:cloudprovenance:cprovl:1.0:attribute-category:cprov"
216                       DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression">
217
218                       <xsl:value-of select=" child::cprovl:ID" />
219                     </AttributeValue>
220
221                     <AttributeDesignator MustBePresent="false"
222                       AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
223                       DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression">
224
225                       <!-- Check to see the type of resource -->
226                       <xsl:if test="$fieldType='cprovd:Action'">
227                        <xsl:attribute name="Category">
228                                  <xsl:text>urn:oasis:names:tc:xacml:3.0:attribute-category:action</xsl:text>
229                                 </xsl:attribute>
230
231                        <xsl:attribute name="AttributeId">
232                                  <xsl:text>urn:oasis:names:tc:xacml:3.0:action-id</xsl:text>
```

```
233                             </xsl:attribute>
234                          </xsl:if>
235
236                          <xsl:if test="$fieldType='cprovd:Service'">
237                           <xsl:attribute name="Category">
238
                                  ↪    <xsl:text>urn:com:provenance:cloudprovenance:cprovl:1.0:attribute-category:service</xsl:text>
239                             </xsl:attribute>
240
241                           <xsl:attribute name="AttributeId">
242                              <xsl:text>urn:oasis:names:tc:xacml:3.0:service-id</xsl:text>
243                             </xsl:attribute>
244                          </xsl:if>
245
246                          <xsl:if test="$fieldType='cprovd:Resource'">
247                           <xsl:attribute name="Category">
248                              <xsl:text>urn:oasis:names:tc:xacml:3.0:attribute-category:resource</xsl:text>
249                             </xsl:attribute>
250
251                           <xsl:attribute name="AttributeId">
252                              <xsl:text>urn:oasis:names:tc:xacml:3.0:resource-id</xsl:text>
253                             </xsl:attribute>
254                          </xsl:if>
255
256                          <xsl:if test="$fieldType='cprovd:Subject'">
257                           <xsl:attribute name="Category">
258                              <xsl:text>urn:oasis:names:tc:xacml:1.0:subject-category:access-subject</xsl:text>
259                             </xsl:attribute>
260
261                           <xsl:attribute name="AttributeId">
262                              <xsl:text>urn:oasis:names:tc:xacml:3.0:subject-id</xsl:text>
263                             </xsl:attribute>
264                          </xsl:if>
265
266                          <xsl:if test="not(boolean($fieldType))">
267                           <xsl:attribute name="Category">
268                              <xsl:text>urn:com:provenance:cloudprovenance:cprovl:1.0:attribute-category:cprov</xsl:text>
269                             </xsl:attribute>
270
271                           <xsl:attribute name="AttributeId">
272                              <xsl:text>urn:oasis:names:tc:xacml:3.0:cprov-id</xsl:text>
273                             </xsl:attribute>
274                            </xsl:if>
275                       </AttributeDesignator>
276                      </Match>
277                     </AllOf>
278                    </AnyOf>
279                   </xsl:if>
280                  </xsl:for-each>
281               </Target>
282              </xsl:if>
283
284              <!-- Converting cProvl condition to XACML condition -->
285              <xsl:if test="count(cprovl:SUCH-THAT)!=0">
286               <Condition>
287                 <xsl:for-each select="cprovl:SUCH-THAT/cprovl:statements">
288
289                    <xsl:choose>
290                       <xsl:when test="@cprovl:negate='true'">
291                       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:not">
292                              <Apply
293                                 FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
294                                 <xsl:for-each select="*">
295                                    <xsl:variable name="nodeName"
296                                    select="current()/name()"/>
297                                    <xsl:if test="$nodeName = 'cprovl:grouping'">
298                                       <xsl:call-template name="grouping"/>
299                                    </xsl:if>
300                                    <xsl:if test="$nodeName = 'cprovl:operator'">
301                                       <xsl:call-template name="operator"/>
302                                    </xsl:if>
303                                    <xsl:if test="$nodeName = 'cprovl:statement'">
304                                       <xsl:call-template name="statement"/>
305                                    </xsl:if>
306                                 </xsl:for-each>
307                              </Apply>
308                     </Apply>
309                     </xsl:when>
310                     <xsl:otherwise>
311                        <Apply
```

```
312                         FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
313                         <xsl:for-each select="*">
314
315                            <xsl:variable name="nodeName"
316                               select="current()/name()"/>
317                            <xsl:if test="$nodeName = 'cprovl:grouping'">
318                               <xsl:call-template name="grouping"/>
319                            </xsl:if>
320                            <xsl:if test="$nodeName = 'cprovl:operator'">
321                               <xsl:call-template name="operator"/>
322                            </xsl:if>
323                            <xsl:if test="$nodeName = 'cprovl:statement'">
324                               <xsl:call-template name="statement"/>
325                            </xsl:if>
326                         </xsl:for-each>
327                      </Apply>
328
329                   </xsl:otherwise>
330                </xsl:choose>
331             </xsl:for-each>
332
333          </Condition>
334          </xsl:if>
335       </Rule>
336       </xsl:for-each>
337       <!-- <Rule RuleId="rule-default" Effect="Deny"> <Description> : All other
338        requests are denied</Description> </Rule> -->
339    </Policy>
340 </xsl:template>
341
342 <xsl:template name ="statement" match="cprovl:statement" mode="loop">
343
344    <xsl:if test="@cprovl:negate='true'">
345       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:not">
346          <Apply
347             FunctionId="labs:orange:com:function:ext:xpath-provenance-node-match">
348             <AttributeValue
349                DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
350                XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:cprov">
351                <xsl:value-of select="child::node()/@prov:id" />
352             </AttributeValue>
353          </Apply>
354       </Apply>
355    </xsl:if>
356    <!-- Check if the negate is set to false, or the attribute is missing -->
357    <xsl:if
358       test="@cprovl:negate = 'false' or not(boolean(@cprovl:negate))">
359       <Apply
360          FunctionId="labs:orange:com:function:ext:xpath-provenance-node-match">
361          <AttributeValue
362             DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
363             XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:cprov">
364             <xsl:value-of select="child::node()/@prov:id"/>
365          </AttributeValue>
366       </Apply>
367    </xsl:if>
368 </xsl:template>
369
370 <xsl:template name ="operator" match="cprovl:operator" mode= "loop">
371
372    <xsl:if test="@cprovl:c_operator='cprovd:Equal-to'">
373
374       <Apply FunctionId="labs:orange:com:function:ext:xpath-provenance-nodes-match">
375
376          <xsl:for-each select="cprovl:statement">
377             <!--<Apply
378                FunctionId="labs:orange:com:function:ext:xpath-provenance-node-match"> -->
379             <AttributeValue
380                DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
381                XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:cprov">
382                <xsl:value-of select="child::node()/@prov:id" />
383             </AttributeValue>
384          <!--   </Apply> -->
385          </xsl:for-each>
386       </Apply>
387    </xsl:if>
388
389 </xsl:template>
390
391 <xsl:template name="grouping" match="cprovl:grouping" mode="loop">
```

```
392     <xsl:if test="@cprovl:l_operator='cprovd:And'">
393       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
394
395        <xsl:for-each select="cprovl:statement">
396          <Apply
397             FunctionId="labs:orange:com:function:ext:xpath-provenance-node-match">
398           <AttributeValue
399            DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
400            XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:cprov">
401            <xsl:value-of select="child::node()/@prov:id" />
402           </AttributeValue>
403         </Apply>
404        </xsl:for-each>
405        <xsl:for-each select="cprovl:grouping">
406         <xsl:apply-templates select="." mode="loop" />
407        </xsl:for-each>
408       </Apply>
409     </xsl:if>
410
411     <xsl:if test="@cprovl:l_operator='cprovd:Or'">
412       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:or">
413        <xsl:for-each select="cprovl:statement">
414          <Apply
415             FunctionId="labs:orange:com:function:ext:xpath-provenance-node-match">
416           <AttributeValue
417            DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
418            XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:cprov">
419            <xsl:value-of select="child::node()/@prov:id" />
420           </AttributeValue>
421         </Apply>
422        </xsl:for-each>
423        <xsl:for-each select="cprovl:grouping">
424          <xsl:apply-templates select="." mode="loop" />
425        </xsl:for-each>
426       </Apply>
427      </xsl:if>
428
429      <xsl:if test="@cprovl:l_operator='cprovd:Not'">
430        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:not">
431         <xsl:for-each select="cprovl:statement">
432           <Apply
433              FunctionId="labs:orange:com:function:ext:xpath-provenance-node-match">
434            <AttributeValue
435              DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
436              XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:cprov">
437              <xsl:value-of select="child::node()/@prov:id" />
438            </AttributeValue>
439           </Apply>
440         </xsl:for-each>
441         <xsl:for-each select="cprovl:grouping">
442           <xsl:apply-templates select="." mode="loop" />
443         </xsl:for-each>
444        </Apply>
445      </xsl:if>
446
447      <xsl:if test="not (boolean(@cprovl:l_operator))">
448        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
449          <xsl:for-each select="cprovl:statement">
450         <Apply
451            FunctionId="labs:orange:com:function:ext:xpath-provenance-node-match">
452           <AttributeValue
453            DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
454            XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:cprov">
455              <xsl:value-of select="child::node()/@prov:id" />
456           </AttributeValue>
457         </Apply>
458        </xsl:for-each>
459         <xsl:for-each select="cprovl:grouping">
460           <xsl:apply-templates select="." mode="loop" />
461         </xsl:for-each>
462        </Apply>
463      </xsl:if>
464     </xsl:template>
465 </xsl:stylesheet>
```

A.4 cProv to Prov XSLT Transformer

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:ex="http://labs.orange.com/uk/ex#"
    xmlns:cprov="http://labs.orange.com/uk/cprov#"
    xmlns:cprovd="http://labs.orange.com/uk/cprovd#"
    xmlns:confidenshare="http://labs.orange.com/uk/confidenshare#"
    xmlns:prov="http://www.w3.org/ns/prov#"
    xmlns:ns2="http://openprovenance.org/collection#"
    xmlns:ns3="http://openprovenance.org/validation#"
    xmlns:prim="http://openprovenance.org/primitives#"
    xmlns:pc1="http://www.ipaw.info/pc1/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:cprovl="http://labs.orange.com/uk/cprovl#"
    xmlns:r="http://orangelabs.com/r#" xmlns:cu="http://www.w3.org/1999/xhtml/datatypes/"
    xmlns:xd="http://www.oxygenxml.com/ns/doc/xsl"
    version="2.0">

    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>

        <xsl:template match="/">

            <prov:document xmlns:prov="http://www.w3.org/ns/prov#"
                xmlns:ns2="http://openprovenance.org/collection#"
                xmlns:ns3="http://openprovenance.org/validation#"
                xmlns:prim="http://openprovenance.org/primitives#"
                xmlns:pc1="http://www.ipaw.info/pc1/"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema">

                <xsl:for-each select="cprov:traceabilityDocument/*">

                    <xsl:variable name="varElementType"
                        select="name()"/>

                    <xsl:choose>
                        <xsl:when test="$varElementType='cprov:cResource'">
                        <prov:entity>
                            <xsl:apply-templates select="." mode="loop" />
                        </prov:entity>
                    </xsl:when>

                       <xsl:when test="$varElementType='cprov:pResource'">
                        <prov:entity>
                            <xsl:apply-templates select="." mode="loop" />
                        </prov:entity>
                    </xsl:when>

                    <xsl:when  test="$varElementType='cprov:transition'">
                        <prov:entity>
                            <xsl:apply-templates select="." mode="loop" />
                        </prov:entity>
                    </xsl:when>

                    <xsl:when test="$varElementType='cprov:cProcess'">
                        <prov:activity>
                            <xsl:apply-templates select="." mode="loop" />
                        </prov:activity>
                    </xsl:when>

                    <!-- edges -->
                    <!-- wasDerivedFrom specialization -->
                    <xsl:when test="$varElementType='cprov:wasRepresentationOf'">
                        <prov:wasDerivedFrom>
                            <xsl:apply-templates select="." mode="loop" />
                        </prov:wasDerivedFrom>
                    </xsl:when>

                    <xsl:when test="$varElementType='cprov:wasReferenceOf'">
                        <prov:wasDerivedFrom>
                            <xsl:apply-templates select="." mode="loop" />
                        </prov:wasDerivedFrom>
                    </xsl:when>

                <!-- wasInformedBy specialization -->
```

```
76                  <xsl:when test="$varElementType='cprov:wasImplicitlyCalledBy'">
77                      <prov:wasInformedBy>
78                          <xsl:apply-templates select="." mode="loop" />
79                      </prov:wasInformedBy>
80                  </xsl:when>
81
82                  <xsl:when test="$varElementType='cprov:wasExplicitlyCalledBy'">
83                      <prov:wasInformedBy>
84                          <xsl:apply-templates select="." mode="loop" />
85                      </prov:wasInformedBy>
86                  </xsl:when>
87
88                  <xsl:when test="$varElementType='cprov:wasRecurrentlyCalledBy'">
89                      <prov:wasInformedBy>
90                          <xsl:apply-templates select="." mode="loop" />
91                      </prov:wasInformedBy>
92                  </xsl:when>
93
94                  <!-- Specialization of wasGeneratedBy -->
95                  <xsl:when test="$varElementType='cprov:hadTransitionState_a'">
96                      <prov:wasDerivedFrom>
97                          <xsl:apply-templates select="." mode="loop" />
98                      </prov:wasDerivedFrom>
99                  </xsl:when>
100
101                     <xsl:when test="$varElementType='cprov:hadTransitionState_b'">
102                         <prov:wasGeneratedBy>
103                             <xsl:apply-templates select="." mode="loop" />
104                         </prov:wasGeneratedBy>
105                     </xsl:when>
106
107                     <xsl:when test="$varElementType='cprov:hadTransitionState_c'">
108                         <prov:wasAttributedTo>
109                             <xsl:apply-templates select="." mode="loop" />
110                         </prov:wasAttributedTo>
111                     </xsl:when>
112
113                 <xsl:when test="$varElementType='cprov:wasVirtualizedBy'">
114                     <prov:wasGeneratedBy>
115                         <xsl:apply-templates select="." mode="loop" />
116                     </prov:wasGeneratedBy>
117                 </xsl:when>
118
119                 <!-- Specialization of wasAssociatedWith -->
120                 <xsl:when test="$varElementType='cprov:wasInitiallyCalledBy'">
121                     <prov:wasAssociatedWith>
122                         <xsl:apply-templates select="." mode="loop" />
123                     </prov:wasAssociatedWith>
124                 </xsl:when>
125
126                 <!-- Specialization of wasAttributedTo -->
127             <xsl:when test="$varElementType='cprov:hadOwnership'">
128                 <prov:wasAttributedTo>
129                     <xsl:apply-templates select="." mode="loop"/>
130                 </prov:wasAttributedTo>
131             </xsl:when>
132             <xsl:otherwise>
133                     <xsl:copy-of  copy-namespaces="no" select=" node()/parent::node()"/>
134             </xsl:otherwise>
135                 </xsl:choose>
136             </xsl:for-each>
137
138         </prov:document>
139     </xsl:template>
140
141 <xsl:template name="conversion" match="node()" mode="loop">
142
143     <xsl:attribute name="prov:id">
144         <xsl:value-of select="@prov:id"></xsl:value-of>
145     </xsl:attribute>
146     <prov:type><xsl:value-of select="name()"></xsl:value-of></prov:type>
147         <xsl:copy-of  copy-namespaces="no" select=" child::node()"/>
148
149     </xsl:template>
150 </xsl:stylesheet>
```

A.5 cProv Policy Request to XACML policy Request XSLT Transformer

```
<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:cprov="http://labs.orange.com/uk/cprov#"
    xmlns:cprovl="http://labs.orange.com/uk/cprovl#" xmlns:prov="http://www.w3.org/ns/prov#"
    xmlns:cprovd="http://labs.orange.com/uk/cprovd#" xmlns:r="http://labs.orange.com/uk/r#"
    xmlns:cu="http://www.w3.org/1999/xhtml/datatypes/" xmlns:ex="http://labs.orange.com/uk/r#"
    xmlns:xd="http://www.oxygenxml.com/ns/doc/xsl" exclude-result-prefixes="xs xd" version="2.0">
    <xd:doc scope="stylesheet">
        <xd:desc>
            <xd:p><xd:b>Updated on:</xd:b> Feb 12, 2013</xd:p>
            <xd:p><xd:b>Author:</xd:b> Mufajjul Ali </xd:p>
            <xd:p/>
        </xd:desc>
    </xd:doc>

    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>

    <xsl:template match="/">
        <Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
             ↪    http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
            ReturnPolicyIdList="true" CombinedDecision="false">

            <xsl:for-each select="cprovl:PolicyRequest/cprovl:Agent">
                <xsl:variable name="fieldRef">
                    <xsl:value-of select=" current()/@isRef"/>
                </xsl:variable>
                <xsl:choose>
                    <!-- Check if it is reference variable -->
                    <xsl:when test="$fieldRef='true'">
                        <Attributes
                            Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
                            <Attribute IncludeInResult="true"
                                AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id">
                                <xsl:attribute name="AttributeId">
                                    <xsl:text>urn:oasis:names:tc:xacml:3.0:subject-id</xsl:text>
                                </xsl:attribute>
                              <!--  <xsl:for-each select="cprovl:PolicyRequest/cprovl:Agent/cprovl:categoryValue"> -->
                                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                                        <xsl:value-of select="@prov:id"/>
                                    </AttributeValue>
                             </Attribute>
                          <!--   </xsl:for-each> -->
                        </Attributes>
                    </xsl:when>
                    <xsl:otherwise>
                        <Attributes
                            Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
                            <Attribute IncludeInResult="true"
                                AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id">
                                <xsl:attribute name="AttributeId">
                                    <xsl:text>urn:oasis:names:tc:xacml:3.0:subject-id</xsl:text>
                                </xsl:attribute>

                                <AttributeValue
                                    DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
                                    XPathCategory="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
                                    <xsl:value-of select="@prov:id"/>
                                </AttributeValue>
                            </Attribute>
                        </Attributes>
                    </xsl:otherwise>
                </xsl:choose>

            </xsl:for-each>
            <xsl:for-each select="cprovl:PolicyRequest/cprovl:Entity">
                <xsl:variable name="requestField">
                    <xsl:value-of select="cprovl:categoryType/text()"/>
                </xsl:variable>

                <xsl:variable name="fieldRef">
```

```
73                  <xsl:value-of select="cprovl:categoryValue/@isRef"/>
74              </xsl:variable>
75
76
77              <xsl:if test="$requestField='cprovd:Action'">
78
79                  <xsl:choose>
80                      <!-- Check if it is reference variable -->
81                      <xsl:when test="$fieldRef='true'">
82
83                          <Attributes
84                              Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
85                              <Attribute IncludeInResult="true"
86                                  AttributeId="urn:oasis:names:tc:xacml:3.0:action-id">
87                                  <xsl:attribute name="AttributeId">
88                                      <xsl:text>urn:oasis:names:tc:xacml:3.0:action-id</xsl:text>
89                                  </xsl:attribute>
90                                  <AttributeValue
91                                      DataType="http://www.w3.org/2001/XMLSchema#string"
92                                      Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
93                                      <xsl:value-of select="cprovl:categoryValue/text()"/>
94                                  </AttributeValue>
95                              </Attribute>
96                          </Attributes>
97                      </xsl:when>
98                      <xsl:otherwise>
99                          <!-- Set it as a xpath value -->
100                         <Attributes
101                             Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
102                             <Attribute IncludeInResult="true"
103                                 AttributeId="urn:oasis:names:tc:xacml:3.0:action-id">
104                                 <xsl:attribute name="AttributeId">
105                                     <xsl:text>urn:oasis:names:tc:xacml:3.0:action-id</xsl:text>
106                                 </xsl:attribute>
107                                 <AttributeValue
108                                     DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
109                                     XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
110                                     <xsl:value-of select="cprovl:categoryValue/text()"/>
111                                 </AttributeValue>
112                             </Attribute>
113                         </Attributes>
114                     </xsl:otherwise>
115                 </xsl:choose>
116             </xsl:if>
117
118             <xsl:if test="$requestField='cprovd:Resource'">
119
120                 <xsl:variable name="fieldRef">
121                     <xsl:value-of select="cprovl:categoryValue/@isRef"/>
122                 </xsl:variable>
123
124                 <xsl:choose>
125                     <!-- Check if it is reference variable -->
126                     <xsl:when test="$fieldRef='true'">
127
128                         <Attributes
129                             Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
130                             <Attribute IncludeInResult="true"
131                                 AttributeId="urn:oasis:names:tc:xacml:3.0:resource-id">
132                                 <xsl:attribute name="AttributeId">
133                                     <xsl:text>urn:oasis:names:tc:xacml:3.0:resource-id</xsl:text>
134                                 </xsl:attribute>
135                                 <AttributeValue
136                                     DataType="http://www.w3.org/2001/XMLSchema#string"
137                                     Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
138                                     <xsl:value-of select="cprovl:categoryValue/text()"/>
139                                 </AttributeValue>
140                             </Attribute>
141                         </Attributes>
142
143                     </xsl:when>
144                     <xsl:otherwise>
145
146                         <Attributes
147                             Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
148                             <Attribute IncludeInResult="true"
149                                 AttributeId="urn:oasis:names:tc:xacml:3.0:resource-id">
150                                 <xsl:attribute name="AttributeId">
151                                     <xsl:text>urn:oasis:names:tc:xacml:3.0:resource-id</xsl:text>
152                                 </xsl:attribute>
```

```
153                                 <AttributeValue
154                                     DataType="http://www.w3.org/2001/XMLSchema#string"
155                                     XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
156                                     <xsl:value-of select="cprovl:categoryValue/text()"/>
157                                 </AttributeValue>
158                             </Attribute>
159                         </Attributes>
160 
161                     </xsl:otherwise>
162 
163                 </xsl:choose>
164 
165 
166             </xsl:if>
167 
168             <!-- TODO: - Update this with ref and normal category -->
169             <xsl:if test="$requestField='cprovd:Service'">
170                 <Attributes
171                     Category="urn:com:provenance:cloudprovenance:cprovl:1.0:attribute-category:service">
172                     <Attribute IncludeInResult="true"
173                         AttributeId="urn:com:provenance:cloudprovenance:cprovl:1.0:service:">
174                         <xsl:attribute name="AttributeId">
175                             <xsl:text>urn:com:provenance:cloudprovenance:cprovl:1.0:service:service-id</xsl:text>
176                         </xsl:attribute>
177                         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
178                             <xsl:value-of select="cprovl:categoryValue/text()"/>
179                         </AttributeValue>
180                     </Attribute>
181                 </Attributes>
182             </xsl:if>
183         </xsl:for-each>
184 
185     </Request>
186 </xsl:template>
187 </xsl:stylesheet>
```

A.6 XACML Policy Response to cProvl policy Response XSLT Transformer

```
<?xml version="1.0" encoding="UTF-8"?>

<!--Author: Mufy -->

<xsl:stylesheet xpath-default-namespace = "urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:cprov="http://labs.orange.com/uk/cprov#"
        xmlns:cprovl="http://labs.orange.com/uk/cprovl#" xmlns:prov="http://www.w3.org/ns/prov#"
        xmlns:cprovd="http://labs.orange.com/uk/cprovd#" xmlns:r="http://labs.orange.com/uk/r#"
        xmlns:cu="http://www.w3.org/1999/xhtml/datatypes/" xmlns:ex="http://labs.orange.com/uk/r#"
        xmlns:xd="http://www.oxygenxml.com/ns/doc/xsl" exclude-result-prefixes="xs xd" version="2.0">
        <xd:doc scope="stylesheet">
            <xd:desc>
                <xd:p><xd:b>Updated on:</xd:b> Feb 12, 2013</xd:p>
                <xd:p><xd:b>Author:</xd:b> Mufajjul Ali </xd:p>
                <xd:p/>
            </xd:desc>
        </xd:doc>

        <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>

        <xsl:template match="/">
            <cprovl:policyResponse xmlns:prov="http://www.w3.org/ns/prov#"
                xmlns:cprov="http://labs.orange.com/uk/cprov#"
                xmlns:opmx="http://openprovenance.org/model/opmx#"
                xmlns:cprovd="http://labs.orange.com/uk/cprovd#"
                xmlns:cu="http://www.w3.org/1999/xhtml/datatypes/"
                xmlns:cprovl="http://labs.orange.com/uk/cprovl#"
                xmlns:r="http://labs.orange.com/uk/r#"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:schemaLocation="http://labs.orange.com/uk/cprovl#
                 ↪    file:/Users/mufy/Dropbox/EngD/year3/schema/cProvl/cProvl-v1.1.xsd">

                <!-- request Id -->
                <!-- decision -->

                <xsl:for-each select="Response/Result">
                    <cprovl:Entity id = 'confidenshare:resultId01'>
                        <cprovl:service>cprovl:confidenshare</cprovl:service>
                        <cprovl:decision><xsl:value-of select="Decision"/></cprovl:decision>
                        <cprovl:description><xsl:value-of select="Status/StatusCode/@Value"/></cprovl:description>
                    </cprovl:Entity>
                </xsl:for-each>

                <xsl:for-each select="Response/Result/Attributes/Attribute">
                    <xsl:if test="@AttributeId='urn:oasis:names:tc:xacml:3.0:subject-id'">

                        <cprovl:Agent prov:id ="ex:001">
                        <xsl:attribute name="prov:id">
                            <xsl:value-of select="AttributeValue"></xsl:value-of>
                        </xsl:attribute>
                    </cprovl:Agent>
                    </xsl:if>

                    <xsl:if test="@AttributeId='urn:oasis:names:tc:xacml:3.0:action-id'">

                        <cprovl:cProcess prov:id ="ex:001">
                            <xsl:attribute name="prov:id">
                                <xsl:value-of select="AttributeValue"></xsl:value-of>
                            </xsl:attribute>
                        </cprovl:cProcess>

                    </xsl:if>

                    <xsl:if test="@AttributeId='urn:oasis:names:tc:xacml:3.0:resource-id'">

                        <cprovl:cResource prov:id ="ex:001">
                            <xsl:attribute name="prov:id">
                                <xsl:value-of select="AttributeValue"></xsl:value-of>
                            </xsl:attribute>
                        </cprovl:cResource>

                    </xsl:if>
```

```
73
74              </xsl:for-each>
75
76
77
78              <!-- processes for target and conditional statements -->
79             <!--   <cprov:cProcess/> -->
80              <cprov:transition/>
81              <!-- edges -->
82              <cprov:hadOwnership/>
83              <cprov:wasReferenceOf/>
84              <cprov:wasImplicitlyCalledBy/>
85              <cprov:wasImplicitlyCalledBy/>
86              <cprov:wasVirtualizedBy/>
87              <cprov:wasVirtualizedBy/>
88              <cprov:wasInitiallyCalledBy/>
89              <prov:used/>
90              <cprov:hadTransitionState_c/>
91
92           </cprovl:policyResponse>
93
94       </xsl:template>
95
96 </xsl:stylesheet>
```

A.7 Service Requirements

A.7.1 Policy 1

```
<?xml version="1.0" encoding="UTF-8"?>
<cprovl:POLICY xmlns:cprov="http://labs.orange.com/uk/cprov#" xmlns:cprovl="http://labs.orange.com/uk/cprovl#"
        xmlns:prov="http://www.w3.org/ns/prov#" xmlns:cprovd="http://labs.orange.com/uk/cprovd#"
        xmlns:r="http://labs.orange.com/uk/r#" xmlns:cu="http://www.w3.org/1999/xhtml/datatypes/"
        xmlns:confidenshare="http://labs.orange.com/uk/confidenshare#" xmlns:opmx="http://openprovenance.org/model/opmx#"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://labs.orange.com/uk/cprovl#
        file:/Users/mufy/Dropbox/EngD/year3/schema/cProvl/cProvl-v1.1.xsd">

        <cprovl:entity prov:id="confidenshare:policy1">
                <cprovl:description>
                        A file (fileA) can only be shared (fileA) by a registered user (userA) to another
                        user (userB)
                </cprovl:description>
        </cprovl:entity>
        <cprovl:ENV>
                <cprovl:entity prov:id="confidenshare:reg.share.confidenshare.labs.orange.com">
                        <cprovl:description />
                </cprovl:entity>
        </cprovl:ENV>

        <cprovl:RULE>
                <cprovl:entity prov:id="confidenshare:rule1">
                        <cprovl:description>
                        </cprovl:description>
                </cprovl:entity>

                <cprovl:DEF>
                        <cprovl:entity prov:id="confidenshare:def">
                                <cprovl:range>cprovd:All</cprovl:range>
                        </cprovl:entity>
                </cprovl:DEF>
                <cprovl:IF cprovl:quantifier="cprovd:There-exist">

                        <!-- Check for input request user and match it with the store -->
                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:d-ref" cprovl:var_identifier="r:ag001"
                                 ↪    cprovl:category_type="cprovd:Subject">r:ag001</cprovl:ID>
                        </cprovl:target>

                        <!-- Check if process createUser exists-->
                        <cprovl:target>
                                <cprovl:ID cprovl:category_type="cprovd:Action">confidenshare:createUser</cprovl:ID>
                        </cprovl:target>

                </cprovl:IF>
                <cprovl:SUCH-THAT>

                        <cprovl:statements>

                                <!-- check if there is an edge between the user ( r:ag001) and process createUser
                                        -->
                                <cprovl:statement>
                                        <cprov:wasInitiallyCalledBy prov:id="confidenshare:wicb001">
                                                <prov:activity prov:ref="confidenshare:createUser" />
                                                <prov:agent prov:ref="r:ag001" />
                                        </cprov:wasInitiallyCalledBy>
                                </cprovl:statement>

                                <!-- check the outcome if entity (ex:createdUser) has been created successfully -->
                                <cprovl:statement>
                                        <cprov:hadOwnership prov:id="confidenshare:hd1">
                                                <prov:entity prov:ref="confidenshare:createdSuccessfully" />
                                                <prov:agent prov:ref="r:ag001" />
                                                <cprov:ownershipType>cprovd:originator</cprov:ownershipType>
                                        </cprov:hadOwnership>
                                </cprovl:statement>

                        </cprovl:statements>
                </cprovl:SUCH-THAT>
                <cprovl:THEN>
                        <cprovl:entity prov:id="confidenshare:result">
                                <cprovl:actionType>cprovd:Permit</cprovl:actionType>
```

```
72                              <cprovl:resourceId>confidenshare:share</cprovl:resourceId>
73                      </cprovl:entity>
74                      <cprov:hadOwnership>
75                              <prov:entity prov:ref="confidenshare:result" />
76                              <prov:agent prov:ref="r:ag001" />
77                              <cprov:ownershipType>cprovd:possession</cprov:ownershipType>
78                      </cprov:hadOwnership>
79              </cprovl:THEN>
80      </cprovl:RULE>
81
82      <cprovl:RULE>
83              <cprovl:entity>
84                      <cprovl:description>Otherwise deny all</cprovl:description>
85              </cprovl:entity>
86              <cprovl:DEF>
87                      <cprovl:entity>
88                              <cprovl:range>cprovd:All</cprovl:range>
89                      </cprovl:entity>
90              </cprovl:DEF>
91              <cprovl:IF cprovl:quantifier="cprovd:There-exist">
92                      <cprovl:target>
93                              <cprovl:ID cprovl:var_type="cprovd:s-ref" cprovl:var_identifier="r:ag001"
                                  ↪    cprovl:category_type="cprovd:Subject">r:ag001</cprovl:ID>
94                      </cprovl:target>
95              </cprovl:IF>
96              <cprovl:THEN>
97                      <cprovl:entity prov:id="confidenshare:result">
98                              <cprovl:actionType>cprovd:Deny</cprovl:actionType>
99                              <cprovl:resourceId>confidenshare:share</cprovl:resourceId>
100                     </cprovl:entity>
101                     <cprov:hadOwnership>
102                             <prov:entity prov:ref="confidenshare:result" />
103                             <prov:agent prov:ref="r:ag001" />
104                             <cprov:ownershipType>cprovd:possession</cprov:ownershipType>
105                     </cprov:hadOwnership>
106             </cprovl:THEN>
107     </cprovl:RULE>
108
109 </cprovl:POLICY>
```

A.7.2 Traceability Graph for Policy 1

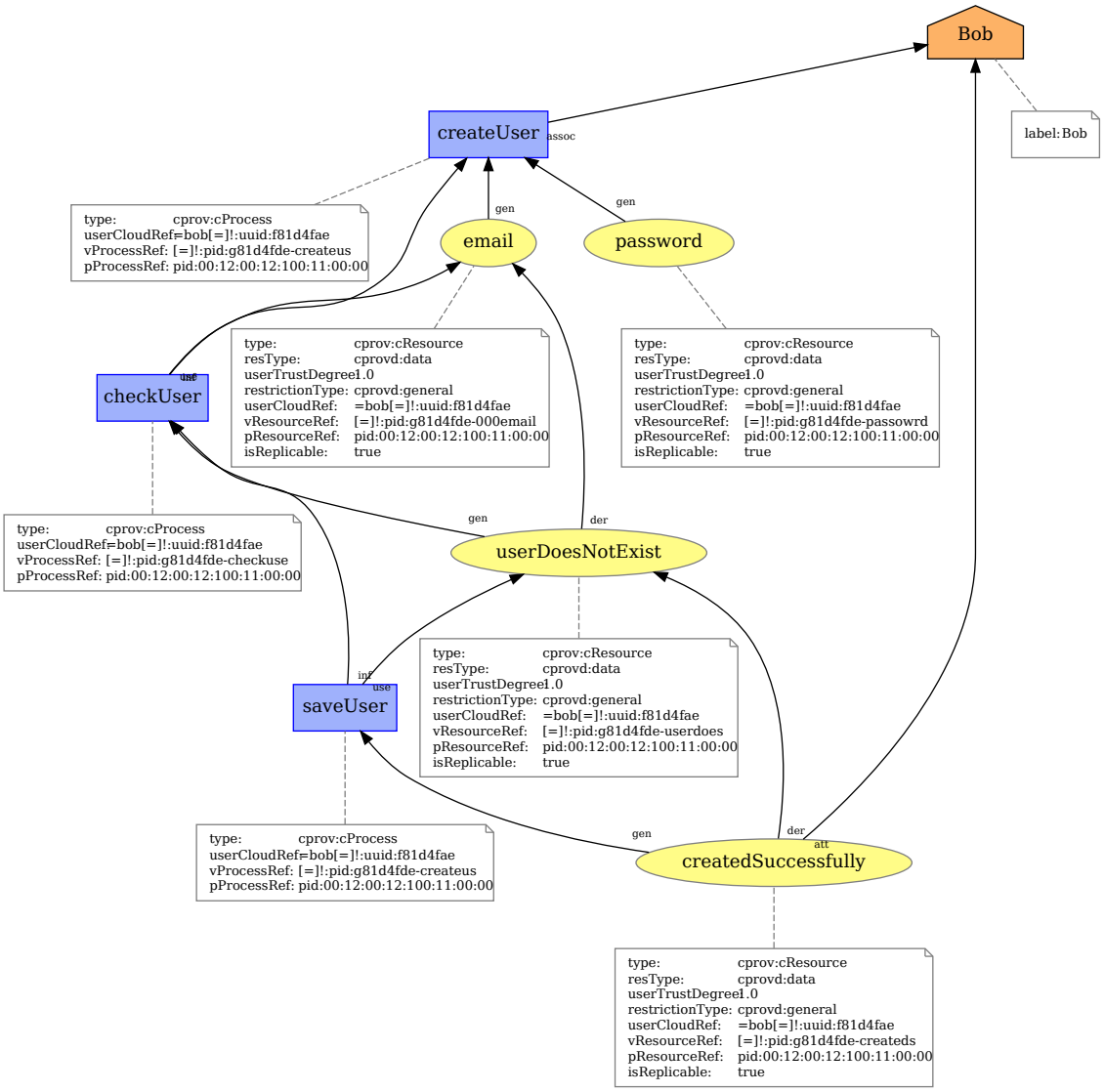


FIGURE A.1: Traceability Graph 1

A.7.3 Policy 2

```
<?xml version="1.0" encoding="UTF-8"?>
<cprovl:POLICY xmlns:cprov="http://labs.orange.com/uk/cprov#" xmlns:cprovl="http://labs.orange.com/uk/cprovl#"
        xmlns:prov="http://www.w3.org/ns/prov#" xmlns:cprovd="http://labs.orange.com/uk/cprovd#"
        xmlns:r="http://labs.orange.com/uk/r#" xmlns:cu="http://www.w3.org/1999/xhtml/datatypes/"
        xmlns:confidenshare="http://labs.orange.com/uk/confidenshare#" xmlns:opmx="http://openprovenance.org/model/opmx#"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://labs.orange.com/uk/cprovl#
        file:/Users/mufy/Dropbox/EngD/year3/schema/cProvl/cProvl-v1.1.xsd">

        <cprovl:entity prov:id="confidenshare:policy2">
                <cprovl:description>
                        If a file (fileA) is marked as confidential, only the originator is allowed to share
                        it with another user (userB), re-sharing by the receiver (userB) to another user
                        (userC) is not allowed
                </cprovl:description>
        </cprovl:entity>

        <cprovl:ENV>
                <cprovl:entity prov:id="confidenshare:confidential.share.confidenshare.labs.orange.com">
                        <cprovl:description />
                </cprovl:entity>
        </cprovl:ENV>

        <cprovl:RULE>
                <cprovl:entity prov:id="confidenshare:rule1">
                        <cprovl:description>
                        </cprovl:description>
                </cprovl:entity>
                <cprovl:DEF>
                        <cprovl:entity prov:id="confidenshare:def">
                                <cprovl:range>cprovd:All</cprovl:range>
                        </cprovl:entity>
                </cprovl:DEF>
                <cprovl:IF cprovl:quantifier="cprovd:There-exist">

                        <!-- Check for input resource against the store -->
                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:d-ref" cprovl:var_identifier="r:e001"
                                 ↪    cprovl:category_type="cprovd:Resource">r:e001</cprovl:ID>
                        </cprovl:target>

                        <!-- Check for the request user -->
                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:d-ref" cprovl:var_identifier="r:ag001"
                                 ↪    cprovl:category_type="cprovd:Subject">r:ag001</cprovl:ID>
                        </cprovl:target>

                        <!-- Process for invoking of the sharing process -->
                        <cprovl:target>
                                <cprovl:ID cprovl:category_type="cprovd:Action">confidenshare:share</cprovl:ID>
                        </cprovl:target>

                </cprovl:IF>
                <cprovl:SUCH-THAT>
                        <cprovl:statements>

                                <!-- Check if the file has confidential restriction -->
                                <cprovl:statement>
                                        <cprov:cResource prov:id="r:e001">
                                                <cprov:restrictionType>cprovd:confidential</cprov:restrictionType>
                                        </cprov:cResource>
                                </cprovl:statement>

                                <!-- Check if the user is not the originator of the resource -->
                                <cprovl:statement cprovl:negate="false">
                                        <cprov:hadOwnership prov:id="confidenshare:hd1">
                                                <prov:entity prov:ref="r:e001" />
                                                <prov:agent prov:ref="r:ag001" />
                                                <cprov:ownershipType>cprovd:originator</cprov:ownershipType>
                                        </cprov:hadOwnership>
                                </cprovl:statement>

                        </cprovl:statements>
                </cprovl:SUCH-THAT>
                <cprovl:THEN>
                        <cprovl:entity prov:id="confidenshare:result">
                                <cprovl:actionType>cprovd:Permit</cprovl:actionType>
```

```
75                              <cprovl:resourceId>confidenshare:share</cprovl:resourceId>
76                      </cprovl:entity>
77                      <cprov:hadOwnership>
78                              <prov:entity prov:ref="confidenshare:result" />
79                              <prov:agent prov:ref="r:ag001" />
80                              <cprov:ownershipType>cprovd:possession</cprov:ownershipType>
81                      </cprov:hadOwnership>
82              </cprovl:THEN>
83      </cprovl:RULE>
84
85      <cprovl:RULE>
86              <cprovl:entity>
87                      <cprovl:description>Otherwise deny all</cprovl:description>
88              </cprovl:entity>
89              <cprovl:DEF>
90                      <cprovl:entity>
91                              <cprovl:range>cprovd:All</cprovl:range>
92                      </cprovl:entity>
93              </cprovl:DEF>
94              <cprovl:IF cprovl:quantifier="cprovd:There-exist">
95                      <cprovl:target>
96                                      <cprovl:ID cprovl:var_type="cprovd:s-ref" cprovl:var_identifier="r:ag001"
                                         ↪    cprovl:category_type="cprovd:Subject">r:ag001</cprovl:ID>
97              </cprovl:target>
98              </cprovl:IF>
99              <cprovl:THEN>
100                     <cprovl:entity prov:id="confidenshare:result">
101                             <cprovl:actionType>cprovd:Deny</cprovl:actionType>
102                             <cprovl:resourceId>confidenshare:share</cprovl:resourceId>
103                     </cprovl:entity>
104                     <cprov:hadOwnership>
105                             <prov:entity prov:ref="confidenshare:result" />
106                             <prov:agent prov:ref="r:ag001" />
107                             <cprov:ownershipType>cprovd:possession</cprov:ownershipType>
108                     </cprov:hadOwnership>
109             </cprovl:THEN>
110     </cprovl:RULE>
111 </cprovl:POLICY>
```

A.7.4 Traceability Graph for Policy 2

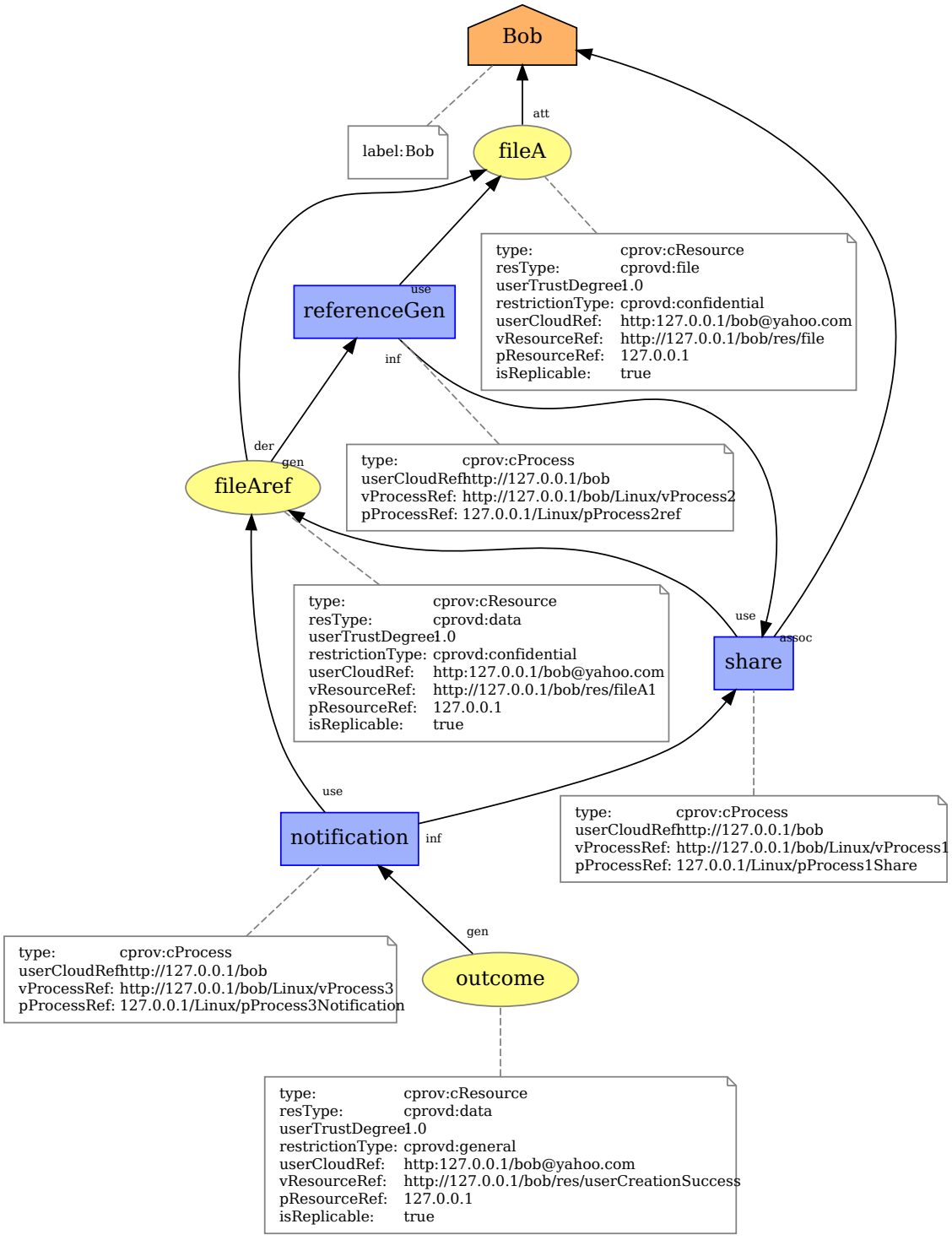


FIGURE A.2: Traceability Graph 2

A.7.5 policy 3

```
<?xml version="1.0" encoding="UTF-8"?>
<cprovl:POLICY xmlns:cprov="http://labs.orange.com/uk/cprov#" xmlns:cprovl="http://labs.orange.com/uk/cprovl#"
        xmlns:prov="http://www.w3.org/ns/prov#" xmlns:cprovd="http://labs.orange.com/uk/cprovd#"
        xmlns:r="http://labs.orange.com/uk/r#" xmlns:cu="http://www.w3.org/1999/xhtml/datatypes/"
        xmlns:confidenshare="http://labs.orange.com/uk/confidenshare#" xmlns:opmx="http://openprovenance.org/model/opmx#"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://labs.orange.com/uk/cprovl#
        file:/Users/mufy/Dropbox/EngD/year3/schema/cProvl/cProvl-v1.1.xsd">

        <cprovl:entity prov:id="confidenshare:policy3">
                <cprovl:description>
                        If a file (fileA) is categorized as restricted, only the originator (userA) and
                        the receivers (userB ... userN) are allowed to modify the file (fileA and its modifications
                        fileA1 ... fileAn) and share (explicitly re-share) amongst themselves only
                </cprovl:description>
        </cprovl:entity>

        <cprovl:ENV>
                <cprovl:entity prov:id="confidenshare:restricted.share.confidenshare.labs.orange.com">
                        <cprovl:description />
                </cprovl:entity>
        </cprovl:ENV>

        <cprovl:RULE>
                <cprovl:entity prov:id="confidenshare:rule1">
                        <cprovl:description>
                        </cprovl:description>
                </cprovl:entity>
                <cprovl:DEF>
                        <cprovl:entity prov:id="confidenshare:def">
                                <cprovl:range>cprovd:All</cprovl:range>
                        </cprovl:entity>
                </cprovl:DEF>
                <cprovl:IF cprovl:quantifier="cprovd:There-exist">

                        <!-- Match the request resource against the store -->
                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:d-ref" cprovl:var_identifier="r:e001"
                                 ↪    cprovl:category_type="cprovd:Resource">r:e001</cprovl:ID>
                        </cprovl:target>

                        <!-- Check for the request user -->
                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:d-ref" cprovl:var_identifier="r:ag001"
                                 ↪    cprovl:category_type="cprovd:Subject">r:ag001</cprovl:ID>
                        </cprovl:target>

                        <!-- Process for sharing -->
                        <cprovl:target>
                                <cprovl:ID cprovl:category_type="cprovd:Action">confidenshare:share</cprovl:ID>
                        </cprovl:target>

                </cprovl:IF>
                <cprovl:SUCH-THAT>

                        <cprovl:statements>

                                <!-- Check if the file is not confidential -->
                                <cprovl:statement cprovl:negate="true">
                                        <cprov:cResource prov:id="r:e001">
                                                <cprov:restrictionType>cprovd:confidential</cprov:restrictionType>
                                        </cprov:cResource>
                                </cprovl:statement>

                                <!-- Check if the user is either the originator or is under his/her possession (i.e
                                        shared) -->
                                <cprovl:grouping cprovl:l_operator="cprovd:Or">
                                        <cprovl:statement>
                                                <cprov:hadOwnership prov:id="confidenshare:hd1">
                                                        <prov:entity prov:ref="r:e001" />
                                                        <prov:agent prov:ref="r:ag001" />
                                                        <cprov:ownershipType>cprovd:originator</cprov:ownershipType>
                                                        <!-- Should it be possession -->
                                                </cprov:hadOwnership>
                                        </cprovl:statement>

                                        <cprovl:statement>
```

```
75                                          <cprov:hadOwnership prov:id="confidenshare:hd2">
76                                                  <prov:entity prov:ref="r:e001" />
77                                                  <prov:agent prov:ref="r:ag001" />
78                                                  <cprov:ownershipType>cprovd:possession</cprov:ownershipType>
79                                          </cprov:hadOwnership>
80                                  </cprovl:statement>
81
82                          </cprovl:grouping>
83                  </cprovl:statements>
84          </cprovl:SUCH-THAT>
85          <cprovl:THEN>
86                  <cprovl:entity prov:id="confidenshare:result">
87                          <cprovl:actionType>cprovd:Permit</cprovl:actionType>
88                          <cprovl:resourceId>confidenshare:share</cprovl:resourceId>
89                  </cprovl:entity>
90                  <cprov:hadOwnership>
91                          <prov:entity prov:ref="confidenshare:result" />
92                          <prov:agent prov:ref="r:ag001" />
93                          <cprov:ownershipType>cprovd:possession</cprov:ownershipType>
94                  </cprov:hadOwnership>
95          </cprovl:THEN>
96      </cprovl:RULE>
97
98      <cprovl:RULE>
99          <cprovl:entity>
100                 <cprovl:description>Otherwise deny all</cprovl:description>
101         </cprovl:entity>
102         <cprovl:DEF>
103                 <cprovl:entity>
104                         <cprovl:range>cprovd:All</cprovl:range>
105                 </cprovl:entity>
106         </cprovl:DEF>
107         <cprovl:IF cprovl:quantifier="cprovd:There-exist">
108                 <cprovl:target>
109                         <cprovl:ID cprovl:var_type="cprovd:s-ref" cprovl:var_identifier="r:ag001"
                              ↪    cprovl:category_type="cprovd:Subject">r:ag001</cprovl:ID>
110                 </cprovl:target>
111         </cprovl:IF>
112         <cprovl:THEN>
113                 <cprovl:entity prov:id="confidenshare:result">
114                         <cprovl:actionType>cprovd:Deny</cprovl:actionType>
115                         <cprovl:resourceId>confidenshare:share</cprovl:resourceId>
116                 </cprovl:entity>
117                 <cprov:hadOwnership>
118                         <prov:entity prov:ref="confidenshare:result" />
119                         <prov:agent prov:ref="r:ag001" />
120                         <cprov:ownershipType>cprovd:possession</cprov:ownershipType>
121                 </cprov:hadOwnership>
122         </cprovl:THEN>
123     </cprovl:RULE>
124
125 </cprovl:POLICY>
```

A.7.6 Traceability Graph for Policy 3

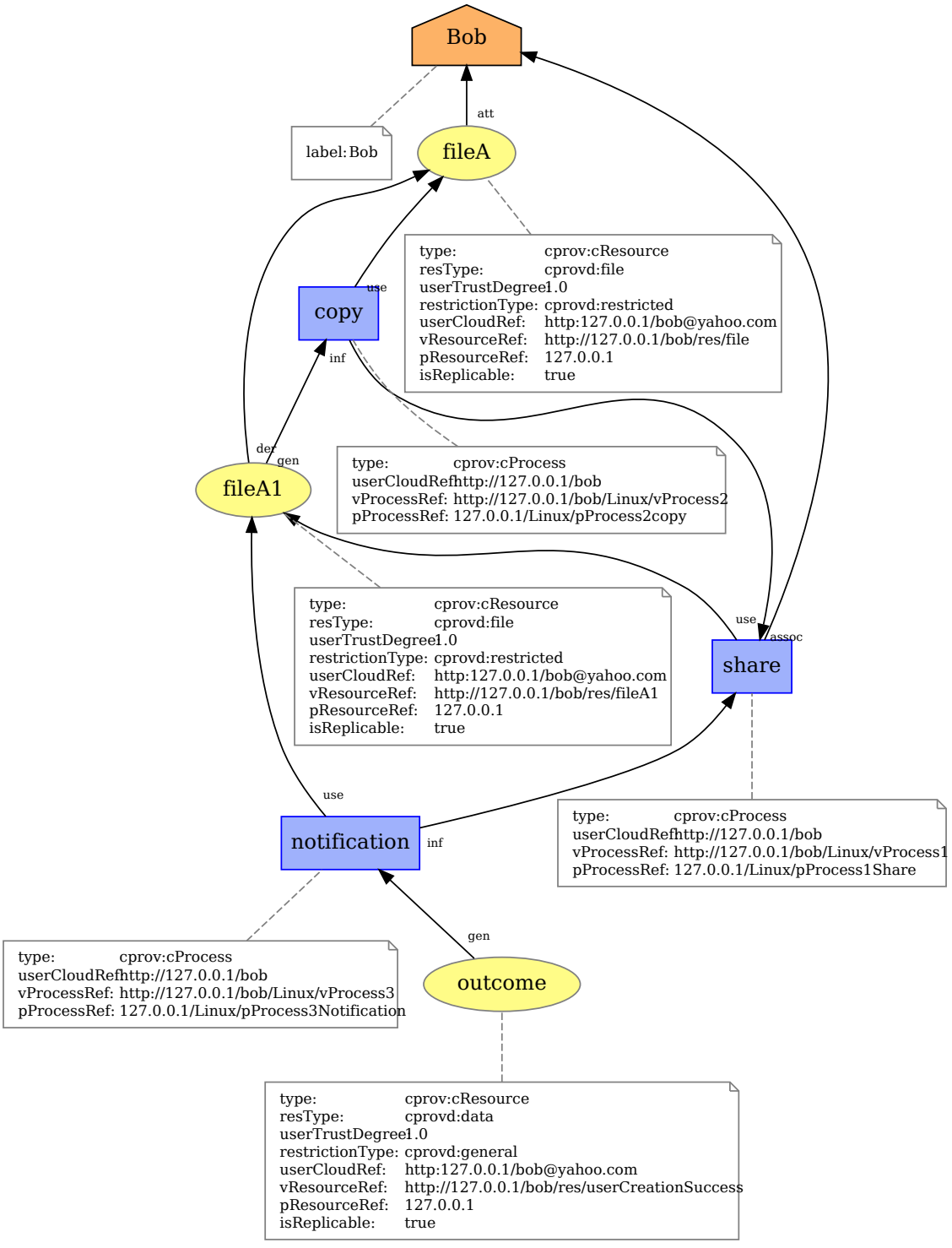


FIGURE A.3: Traceability Graph 3

A.7.7 policy 4

```
<?xml version="1.0" encoding="UTF-8"?>
<cprovl:POLICY xmlns:cprov="http://labs.orange.com/uk/cprov#" xmlns:cprovl="http://labs.orange.com/uk/cprovl#"
        xmlns:prov="http://www.w3.org/ns/prov#" xmlns:cprovd="http://labs.orange.com/uk/cprovd#"
        xmlns:r="http://labs.orange.com/uk/r#" xmlns:cu="http://www.w3.org/1999/xhtml/datatypes/"
        xmlns:confidenshare="http://labs.orange.com/uk/confidenshare#" xmlns:opmx="http://openprovenance.org/model/opmx#"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://labs.orange.com/uk/cprovl#
        file:/Users/mufy/Dropbox/EngD/year3/schema/cProvl/cProvl-v1.1.xsd">

        <cprovl:entity prov:id="confidenshare:policy4">
                <cprovl:description>
                        If a file (fileA) is labelled as general, any user (User A) can share the file with
                        other users (User X)
                </cprovl:description>
        </cprovl:entity>

        <cprovl:ENV>
                <cprovl:entity prov:id="confidenshare:general.share.confidenshare.labs.orange.com">
                        <cprovl:description />
                </cprovl:entity>
        </cprovl:ENV>

        <cprovl:RULE>
                <cprovl:entity prov:id="confidenshare:rule1">
                        <cprovl:description>
                        </cprovl:description>
                </cprovl:entity>
                <cprovl:DEF>
                        <cprovl:entity prov:id="confidenshare:def">
                                <cprovl:range>cprovd:All</cprovl:range>
                        </cprovl:entity>
                </cprovl:DEF>
                <cprovl:IF cprovl:quantifier="cprovd:There-exist">

                        <!-- Match input request resource with the store -->
                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:d-ref" cprovl:var_identifier="r:e001"
                                 ↪    cprovl:category_type="cprovd:Resource">r:e001</cprovl:ID>
                        </cprovl:target>

                        <!-- Check for the request user -->
                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:d-ref" cprovl:var_identifier="r:ag001"
                                 ↪    cprovl:category_type="cprovd:Subject">r:ag001</cprovl:ID>
                        </cprovl:target>

                        <!-- Process for sharing -->
                        <cprovl:target>
                                <cprovl:ID cprovl:category_type="cprovd:Action">confidenshare:share</cprovl:ID>
                        </cprovl:target>

                </cprovl:IF>
                <cprovl:SUCH-THAT>

                        <cprovl:statements>
                                <cprovl:statement>
                                        <cprov:cResource prov:id="r:e001">
                                                <cprov:restrictionType>cprovd:general</cprov:restrictionType>
                                        </cprov:cResource>
                                </cprovl:statement>
                        </cprovl:statements>
                </cprovl:SUCH-THAT>
                <cprovl:THEN>
                        <cprovl:entity prov:id="confidenshare:result">
                                <cprovl:actionType>cprovd:Permit</cprovl:actionType>
                                <cprovl:resourceId>confidenshare:share</cprovl:resourceId>
                        </cprovl:entity>
                        <cprov:hadOwnership>
                                <prov:entity prov:ref="confidenshare:result" />
                                <prov:agent prov:ref="r:ag001" />
                                <cprov:ownershipType>cprovd:possession</cprov:ownershipType>
                        </cprov:hadOwnership>
                </cprovl:THEN>
        </cprovl:RULE>

        <cprovl:RULE>
                <cprovl:entity>
```

```
75                      <cprovl:description>Otherwise deny all</cprovl:description>
76              </cprovl:entity>
77              <cprovl:DEF>
78                      <cprovl:entity>
79                              <cprovl:range>cprovd:All</cprovl:range>
80                      </cprovl:entity>
81              </cprovl:DEF>
82              <cprovl:IF cprovl:quantifier="cprovd:There-exist">
83                      <cprovl:target>
84                              <cprovl:ID cprovl:var_type="cprovd:s-ref" cprovl:var_identifier="r:ag001"
                                 ↪    cprovl:category_type="cprovd:Subject">r:ag001</cprovl:ID>
85                      </cprovl:target>
86              </cprovl:IF>
87              <cprovl:THEN>
88                      <cprovl:entity prov:id="confidenshare:result">
89                              <cprovl:actionType>cprovd:Deny</cprovl:actionType>
90                              <cprovl:resourceId>confidenshare:share</cprovl:resourceId>
91                      </cprovl:entity>
92                      <cprov:hadOwnership>
93                              <prov:entity prov:ref="confidenshare:result" />
94                              <prov:agent prov:ref="r:ag001" />
95                              <cprov:ownershipType>cprovd:possession</cprov:ownershipType>
96                      </cprov:hadOwnership>
97              </cprovl:THEN>
98      </cprovl:RULE>
99 </cprovl:POLICY>
```

A.7.8 Traceability Graph for Policy 4

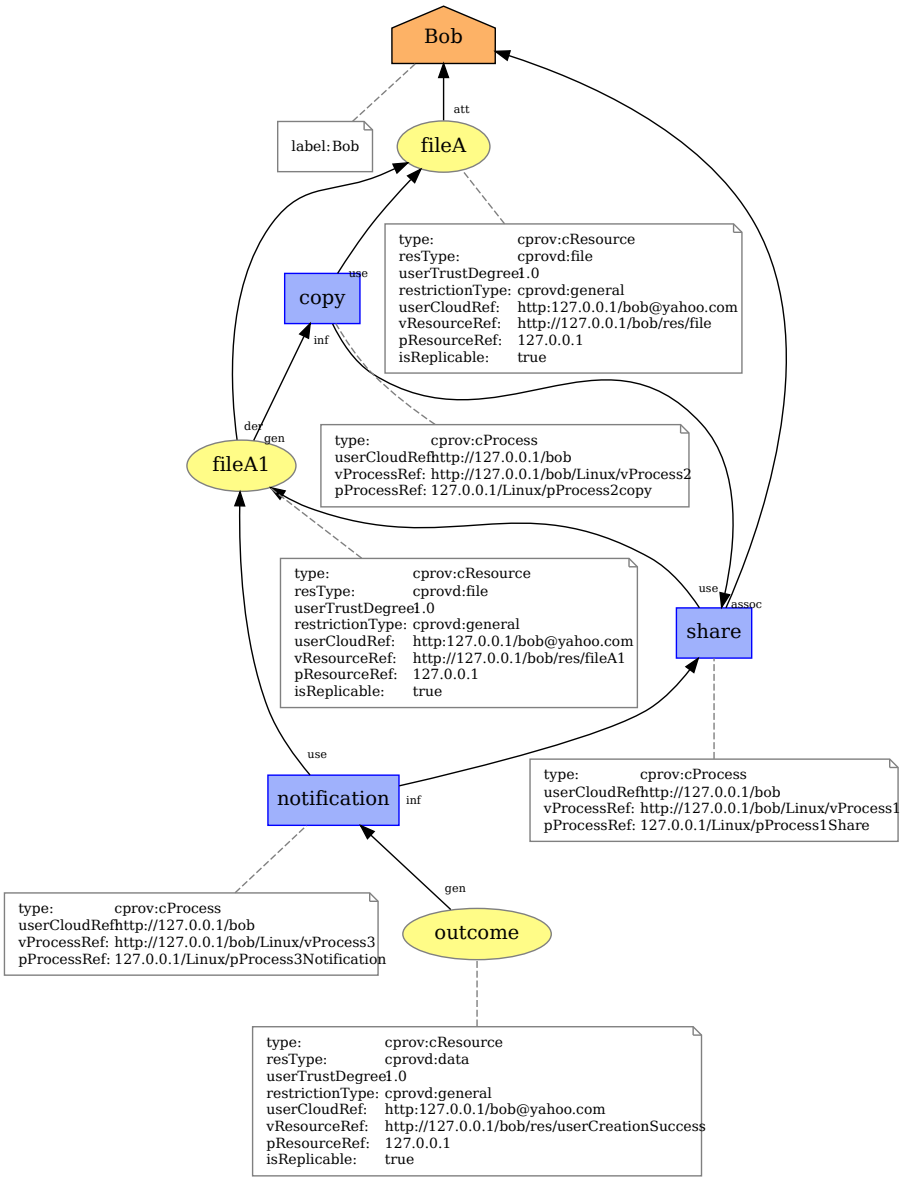


FIGURE A.4: Traceability Graph 4

A.7.9 policy 5

```
<?xml version="1.0" encoding="UTF-8"?>
<cprovl:POLICY xmlns:cprov="http://labs.orange.com/uk/cprov#" xmlns:cprovl="http://labs.orange.com/uk/cprovl#"
        xmlns:prov="http://www.w3.org/ns/prov#" xmlns:cprovd="http://labs.orange.com/uk/cprovd#"
        xmlns:r="http://labs.orange.com/uk/r#" xmlns:cu="http://www.w3.org/1999/xhtml/datatypes/"
        xmlns:confidenshare="http://labs.orange.com/uk/confidenshare#" xmlns:opmx="http://openprovenance.org/model/opmx#"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://labs.orange.com/uk/cprovl#
        file:/Users/mufy/Dropbox/EngD/year3/schema/cProvl/cProvl-v1.1.xsd">

        <cprovl:entity prov:id="confidenshare:policy5">
                <cprovl:description>
                        A user (user A) logged in from one region (EU) cannot share a confidential or restricted
                        file to another user (user B) logged in from a different region (non-EU)
                </cprovl:description>
        </cprovl:entity>

        <cprovl:ENV>
                <cprovl:entity prov:id="confidenshare:region.share.confidenshare.labs.orange.com">
                        <cprovl:description />
                </cprovl:entity>
        </cprovl:ENV>

        <cprovl:RULE>
                <cprovl:entity prov:id="confidenshare:rule1">
                        <cprovl:description>
                        </cprovl:description>
                </cprovl:entity>
                <cprovl:DEF>
                        <cprovl:entity prov:id="confidenshare:def">
                                <cprovl:range>cprovd:All</cprovl:range>
                        </cprovl:entity>
                </cprovl:DEF>
                <cprovl:IF cprovl:quantifier="cprovd:There-exist">

                        <!-- Check input request resource with the store -->
                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:d-ref" cprovl:var_identifier="r:e001"
                                 ↪    cprovl:category_type="cprovd:Resource">r:e001</cprovl:ID>
                        </cprovl:target>

                        <!-- Check for the share user -->
                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:d-ref" cprovl:var_identifier="r:ag001"
                                 ↪    cprovl:category_type="cprovd:Subject">r:ag001</cprovl:ID>
                        </cprovl:target>

                        <!-- Check for the shared with user -->
                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:d-ref" cprovl:var_identifier="r:ag002"
                                 ↪    cprovl:category_type="cprovd:Subject">r:ag002</cprovl:ID>
                        </cprovl:target>

                        <!-- Variable for populating transition details of the both users -->
                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:new"
                                 ↪    cprovl:var_identifier="r:user1Transition">r:user1Transition</cprovl:ID>
                        </cprovl:target>

                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:new"
                                 ↪    cprovl:var_identifier="r:user2Transition">r:user2Transition</cprovl:ID>
                        </cprovl:target>

                        <!-- Process for sharing -->
                        <cprovl:target>
                                <cprovl:ID cprovl:category_type="cprovd:Action">confidenshare:share</cprovl:ID>
                        </cprovl:target>

                </cprovl:IF>
                <cprovl:SUCH-THAT>
                        <cprovl:statements>

                                <!-- check if the resource in not general -->
                                <cprovl:statement cprovl:negate="true">
                                        <cprov:cResource prov:id="r:e001">
                                                <cprov:restrictionType>cprovd:general</cprov:restrictionType>
                                        </cprov:cResource>
```

```
72                              </cprovl:statement>
73
74                              <cprovl:statement>
75                                      <!-- get entity transition for user 1 -->
76                                      <cprov:hadTransitionState_c prov:id="confidenshare:tr1">
77                                              <prov:entity prov:ref="r:user1Transition" />
78                                              <prov:agent prov:ref="r:ag001" />
79                                      </cprov:hadTransitionState_c>
80                              </cprovl:statement>
81
82                              <cprovl:statement>
83                                      <!-- get entity transition for user 2 -->
84                                      <cprov:hadTransitionState_c prov:id="confidenshare:tr2">
85                                              <prov:entity prov:ref="r:user2Transition" />
86                                              <prov:agent prov:ref="r:ag001" />
87                                      </cprov:hadTransitionState_c>
88                              </cprovl:statement>
89
90                              <!-- Check if the both location are the same  -->
91                              <cprovl:operator cprovl:c_operator="cprovd:Equal-to">
92                                      <cprovl:statement>
93                                              <cprov:cResource prov:id="r:user1Transition" />
94                                      </cprovl:statement>
95                                      <cprovl:statement>
96                                              <cprov:cResource prov:id="r:user2Transition" />
97                                      </cprovl:statement>
98                              </cprovl:operator>
99
100                     </cprovl:statements>
101             </cprovl:SUCH-THAT>
102             <cprovl:THEN>
103                     <cprovl:entity prov:id="confidenshare:result">
104                             <cprovl:actionType>cprovd:Permit</cprovl:actionType>
105                             <cprovl:resourceId>confidenshare:share</cprovl:resourceId>
106                     </cprovl:entity>
107                     <cprov:hadOwnership>
108                             <prov:entity prov:ref="confidenshare:result" />
109                             <prov:agent prov:ref="r:ag001" />
110                             <cprov:ownershipType>cprovd:possession</cprov:ownershipType>
111                     </cprov:hadOwnership>
112             </cprovl:THEN>
113     </cprovl:RULE>
114
115     <cprovl:RULE>
116             <cprovl:entity>
117                     <cprovl:description>Otherwise deny all</cprovl:description>
118             </cprovl:entity>
119             <cprovl:DEF>
120                     <cprovl:entity>
121                             <cprovl:range>cprovd:All</cprovl:range>
122                     </cprovl:entity>
123             </cprovl:DEF>
124             <cprovl:IF cprovl:quantifier="cprovd:There-exist">
125                     <cprovl:target>
126                             <cprovl:ID cprovl:var_type="cprovd:s-ref" cprovl:var_identifier="r:ag001"
                                  ↪    cprovl:category_type="cprovd:Subject">r:ag001</cprovl:ID>
127                     </cprovl:target>
128             </cprovl:IF>
129             <cprovl:THEN>
130                     <cprovl:entity prov:id="confidenshare:result">
131                             <cprovl:actionType>cprovd:Deny</cprovl:actionType>
132                             <cprovl:resourceId>confidenshare:share</cprovl:resourceId>
133                     </cprovl:entity>
134                     <cprov:hadOwnership>
135                             <prov:entity prov:ref="confidenshare:result" />
136                             <prov:agent prov:ref="r:ag001" />
137                             <cprov:ownershipType>cprovd:possession</cprov:ownershipType>
138                     </cprov:hadOwnership>
139             </cprovl:THEN>
140     </cprovl:RULE>
141 </cprovl:POLICY>
```

A.7.10 Traceability Graph for Policy 5

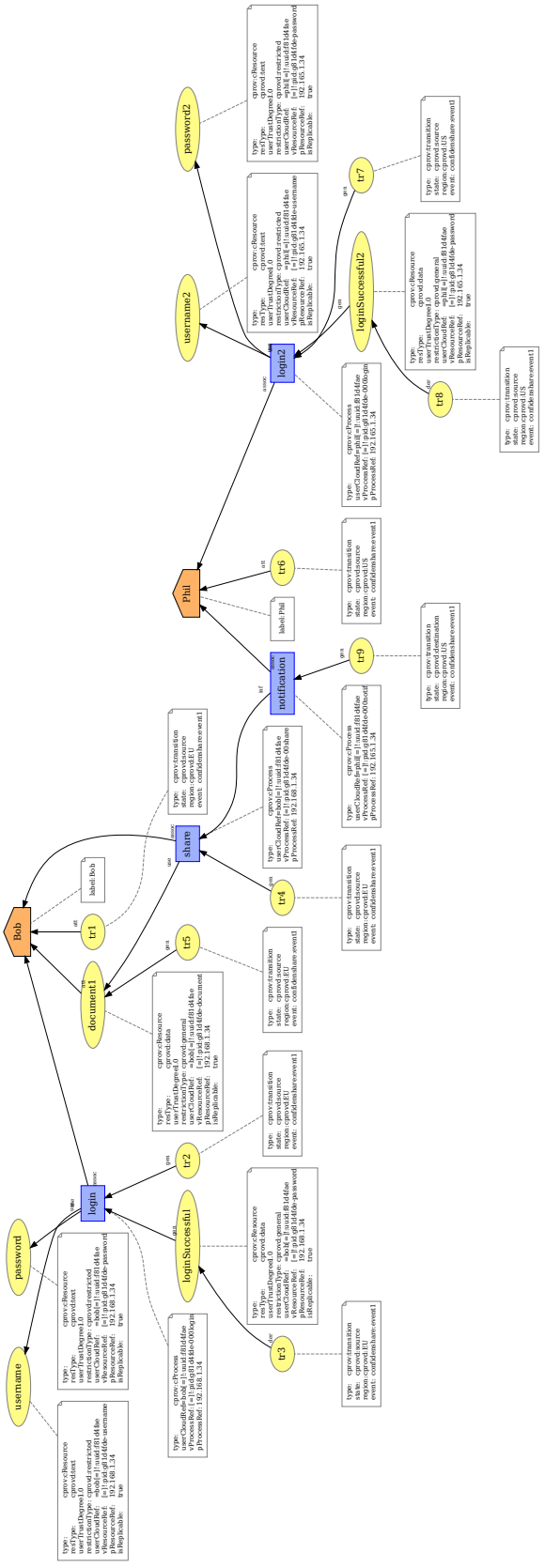


FIGURE A.5: Traceability Graph 5

A.7.11 policy6

```
<?xml version="1.0" encoding="UTF-8"?>
<cprovl:POLICY xmlns:cprov="http://labs.orange.com/uk/cprov#" xmlns:cprovl="http://labs.orange.com/uk/cprovl#"
        xmlns:prov="http://www.w3.org/ns/prov#" xmlns:cprovd="http://labs.orange.com/uk/cprovd#"
        xmlns:r="http://labs.orange.com/uk/r#" xmlns:cu="http://www.w3.org/1999/xhtml/datatypes/"
        xmlns:confidenshare="http://labs.orange.com/uk/confidenshare#" xmlns:opmx="http://openprovenance.org/model/opmx#"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://labs.orange.com/uk/cprovl#
        file:/Users/mufy/Dropbox/EngD/year3/schema/cProvl/cProvl-v1.1.xsd">

        <cprovl:entity prov:id="confidenshare:policy6">
                <cprovl:description>
                        If a user (userA) is classed as removed, all the associated shared files (file X)
                        cannot be shared further
                </cprovl:description>
        </cprovl:entity>

        <cprovl:ENV>
                <cprovl:entity prov:id="confidenshare:removed.share.confidenshare.labs.orange.com">
                        <cprovl:description />
                </cprovl:entity>
        </cprovl:ENV>

        <cprovl:RULE>
                <cprovl:entity prov:id="confidenshare:rule1">
                        <cprovl:description>
                        </cprovl:description>
                </cprovl:entity>
                <cprovl:DEF>
                        <cprovl:entity prov:id="confidenshare:def">
                                <cprovl:range>cprovd:All</cprovl:range>
                        </cprovl:entity>
                </cprovl:DEF>
                <cprovl:IF cprovl:quantifier="cprovd:There-exist">

                        <!-- Check for any input request and match it with the store -->
                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:d-ref" cprovl:var_identifier="r:e001"
                                 ↪    cprovl:category_type="cprovd:Resource">r:e001</cprovl:ID>
                        </cprovl:target>

                        <!-- Check for the request user -->
                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:d-ref" cprovl:var_identifier="r:ag001"
                                 ↪    cprovl:category_type="cprovd:Subject">r:ag001</cprovl:ID>
                        </cprovl:target>

                        <!-- Process for sharing -->
                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:s-ref"
                                 ↪    cprovl:category_type="cprovd:Action">confidenshare:share</cprovl:ID>
                        </cprovl:target>

                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:new"
                                 ↪    cprovl:var_identifier="r:originAgent">r:originAgent</cprovl:ID>
                        </cprovl:target>

                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:new"
                                 ↪    cprovl:var_identifier="r:originFile">r:originFile</cprovl:ID>
                        </cprovl:target>

                </cprovl:IF>
                <cprovl:SUCH-THAT>

                        <cprovl:statements>

                                <!-- Check if user is not the originator of the file -->

                                <cprovl:grouping cprovl:l_operator="cprovd:Or">

                                        <!-- if he/she is the owner of the file, then permit -->
                                        <cprovl:statement>
                                                <cprov:hadOwnership prov:id="confidenshare:hd1">
                                                        <prov:entity prov:ref="r:e001" />
                                                        <prov:agent prov:ref="r:ag001" />
                                                        <cprov:ownershipType>cprovd:Originator</cprov:ownershipType>
```

```
72                                      </cprov:hadOwnership>
73                                  </cprovl:statement>
74
75                                  <!-- otherwsie check for the following conditions for removal share violation-->
76                                  <cprovl:grouping>
77                                      <!-- File is shared with the user (in his/her possession) -->
78                                      <cprovl:statement>
79                                          <cprov:hadOwnership prov:id="confidenshare:hd2">
80                                              <prov:entity prov:ref="r:e001" />
81                                              <prov:agent prov:ref="r:ag001" />
82                                              <cprov:ownershipType>cprovd:Possession</cprov:ownershipType>
83                                          </cprov:hadOwnership>
84                                      </cprovl:statement>
85
86                                      <!-- Check if the file was derived from another file-->
87                                      <cprovl:statement>
88                                          <cprov:wasRepresentationOf prov:id="confidenshare:wro1">
89                                              <prov:generatedEntity prov:ref="r:e001" />
90                                              <prov:usedEntity prov:ref="r:originFile" />
91                                          </cprov:wasRepresentationOf>
92                                      </cprovl:statement>
93
94                                      <!-- get the origin file user -->
95                                      <cprovl:statement>
96                                          <cprov:hadOwnership prov:id="confidenshare:hd3">
97                                              <prov:entity prov:ref="r:originFile" />
98                                              <prov:agent prov:ref="r:originAgent" />
99                                              <cprov:ownershipType>cprovd:Originator</cprov:ownershipType>
100                                         </cprov:hadOwnership>
101                                     </cprovl:statement>
102
103                                     <!-- Check if the origin file owner is deleted -->
104                                     <cprovl:statement>
105                                         <cprov:wasInitiallyCalledBy prov:id="confidenshare:wic1">
106                                             <prov:activity prov:ref="confidenshare:delete" />
107                                             <prov:agent prov:ref="r:originAgent" />
108                                         </cprov:wasInitiallyCalledBy>
109                                     </cprovl:statement>
110                                 </cprovl:grouping>
111                             </cprovl:grouping>
112
113                 </cprovl:statements>
114             </cprovl:SUCH-THAT>
115             <cprovl:THEN>
116                 <cprovl:entity prov:id="confidenshare:result">
117                     <cprovl:actionType>cprovd:Permit</cprovl:actionType>
118                     <cprovl:resourceId>confidenshare:share</cprovl:resourceId>
119                 </cprovl:entity>
120                 <cprov:hadOwnership>
121                     <prov:entity prov:ref="confidenshare:result" />
122                     <prov:agent prov:ref="r:ag001" />
123                     <cprov:ownershipType>cprovd:possession</cprov:ownershipType>
124                 </cprov:hadOwnership>
125             </cprovl:THEN>
126     </cprovl:RULE>
127
128     <cprovl:RULE>
129             <cprovl:entity>
130                 <cprovl:description>Otherwise deny all</cprovl:description>
131             </cprovl:entity>
132             <cprovl:DEF>
133                 <cprovl:entity>
134                     <cprovl:range>cprovd:All</cprovl:range>
135                 </cprovl:entity>
136             </cprovl:DEF>
137             <cprovl:IF cprovl:quantifier="cprovd:There-exist">
138                 <cprovl:target>
139                     <cprovl:ID cprovl:var_type="cprovd:s-ref" cprovl:var_identifier="r:ag001"
                          ↪    cprovl:category_type="cprovd:Subject">r:ag001</cprovl:ID>
140                 </cprovl:target>
141             </cprovl:IF>
142             <cprovl:THEN>
143                 <cprovl:entity prov:id="confidenshare:result">
144                     <cprovl:actionType>cprovd:Deny</cprovl:actionType>
145                     <cprovl:resourceId>confidenshare:share</cprovl:resourceId>
146                 </cprovl:entity>
147                 <cprov:hadOwnership>
148                     <prov:entity prov:ref="confidenshare:result" />
149                     <prov:agent prov:ref="r:ag001" />
150                     <cprov:ownershipType>cprovd:possession</cprov:ownershipType>
```

```
151                     </cprov:hadOwnership>
152                 </cprovl:THEN>
153             </cprovl:RULE>
154
155 </cprovl:POLICY>
```

A.7.12 Traceability Graph for Policy 6

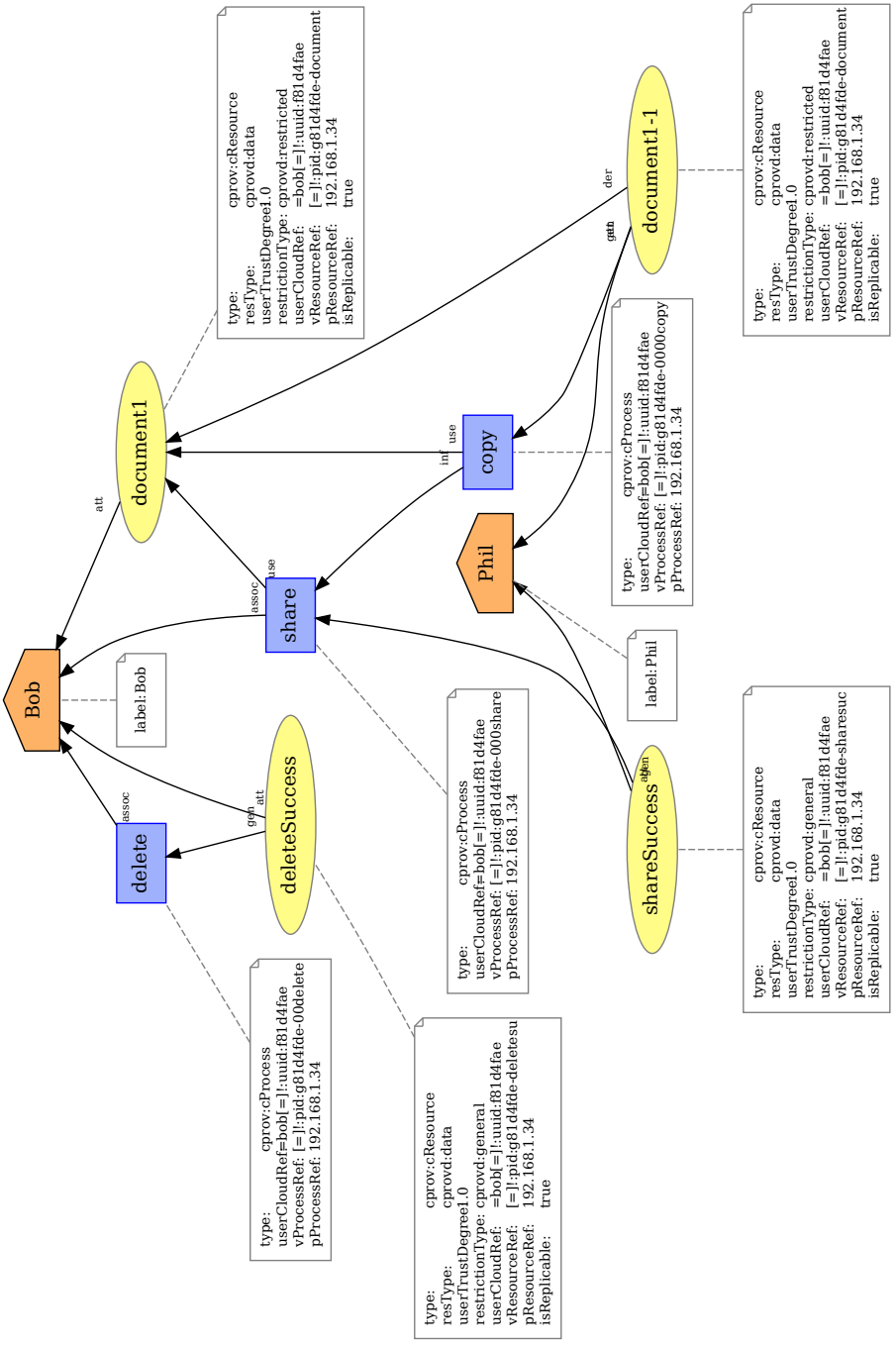


FIGURE A.6: Traceability Graph 6

A.7.13 policy 7

```
<?xml version="1.0" encoding="UTF-8"?>
<cprovl:POLICY xmlns:cprov="http://labs.orange.com/uk/cprov#" xmlns:cprovl="http://labs.orange.com/uk/cprovl#"
        xmlns:prov="http://www.w3.org/ns/prov#" xmlns:cprovd="http://labs.orange.com/uk/cprovd#"
        xmlns:r="http://labs.orange.com/uk/r#" xmlns:cu="http://www.w3.org/1999/xhtml/datatypes/"
        xmlns:confidenshare="http://labs.orange.com/uk/confidenshare#" xmlns:opmx="http://openprovenance.org/model/opmx#"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://labs.orange.com/uk/cprovl#
        file:/Users/mufy/Dropbox/EngD/year3/schema/cProvl/cProvl-v1.1.xsd">

        <cprovl:entity prov:id="confidenshare:policy7">
                <cprovl:description>
                        The storage of a new sensitive file (fileA) needs to reside it the same region as
                        the registration of the user (userA), otherwise it is denied
                </cprovl:description>
        </cprovl:entity>

        <cprovl:ENV>
                <cprovl:entity prov:id="region.store.confidenshare.labs.orange.com">
                        <cprovl:description />
                </cprovl:entity>
        </cprovl:ENV>

        <cprovl:RULE>
                <cprovl:entity prov:id="confidenshare:rule1">
                        <cprovl:description>
                        </cprovl:description>
                </cprovl:entity>
                <cprovl:DEF>
                        <cprovl:entity prov:id="confidenshare:def">
                                <cprovl:range>cprovd:All</cprovl:range>
                        </cprovl:entity>
                </cprovl:DEF>
                <cprovl:IF cprovl:quantifier="cprovd:There-exist">

                        <!-- Check for input request resource against the store -->
                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:d-ref" cprovl:var_identifier="r:e001"
                                 ↪    cprovl:category_type="cprovd:Resource">r:e001</cprovl:ID>
                        </cprovl:target>

                        <!-- Check for the request user -->
                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:d-ref" cprovl:var_identifier="r:ag001"
                                 ↪    cprovl:category_type="cprovd:Subject">r:ag001</cprovl:ID>
                        </cprovl:target>

                        <!-- Process for login -->
                        <cprovl:target>
                                <cprovl:ID  cprovl:category_type="cprovd:Action">confidenshare:login</cprovl:ID>
                        </cprovl:target>

                <!-- Variables for populating transition information -->
                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:new" cprovl:var_identifier="r:docTrans">r:docTrans</cprovl:ID>
                        </cprovl:target>

                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:new"
                                 ↪    cprovl:var_identifier="r:loginTrans">r:loginTrans</cprovl:ID>
                        </cprovl:target>

                </cprovl:IF>
                <cprovl:SUCH-THAT>

                        <cprovl:statements>

                                <!-- Check if the new file is restricted -->
                                <cprovl:statement>
                                        <cprov:cResource prov:id="r:e001">
                                                <cprov:restrictionType>cprovd:restricted</cprov:restrictionType>
                                        </cprov:cResource>
                                </cprovl:statement>

                                <!-- Obtain first location -->
                                <cprovl:statement>
                                        <cprov:hadTransitionState_a prov:id="confidenshare:tr1">
                                                <prov:generatedEntity prov:ref="r:e001"/>
```

```
74                                      <prov:usedEntity prov:ref="r:docTrans"/>
75                              </cprov:hadTransitionState_a>
76                      </cprovl:statement>
77
78                      <!-- Obtain second location -->
79                      <cprovl:statement>
80                              <cprov:hadTransitionState_b prov:id="confidenshare:tr2">
81                                      <prov:activity prov:ref="confidenshare:login"/>
82                                      <prov:entity prov:ref="r:loginTrans"/>
83                              </cprov:hadTransitionState_b>
84                      </cprovl:statement>
85
86                      <!-- Check if the both location are the same  -->
87                      <cprovl:operator cprovl:c_operator="cprovd:Equal-to">
88                              <cprovl:statement>
89                                      <cprov:cResource prov:id="r:docTrans" />
90                              </cprovl:statement>
91                              <cprovl:statement>
92                                      <cprov:cResource prov:id="r:loginTrans" />
93                              </cprovl:statement>
94                      </cprovl:operator>
95
96              </cprovl:statements>
97
98      </cprovl:SUCH-THAT>
99      <cprovl:THEN>
100             <cprovl:entity prov:id="confidenshare:result">
101                     <cprovl:actionType>cprovd:Permit</cprovl:actionType>
102                     <cprovl:resourceId>confidenshare:create</cprovl:resourceId>
103             </cprovl:entity>
104             <cprov:hadOwnership>
105                     <prov:entity prov:ref="confidenshare:result" />
106                     <prov:agent prov:ref="r:ag001" />
107                     <cprov:ownershipType>cprovd:possession</cprov:ownershipType>
108             </cprov:hadOwnership>
109     </cprovl:THEN>
110 </cprovl:RULE>
111
112 <cprovl:RULE>
113     <cprovl:entity>
114             <cprovl:description>Otherwise deny all</cprovl:description>
115     </cprovl:entity>
116     <cprovl:DEF>
117             <cprovl:entity>
118                     <cprovl:range>cprovd:All</cprovl:range>
119             </cprovl:entity>
120     </cprovl:DEF>
121     <cprovl:IF cprovl:quantifier="cprovd:There-exist">
122             <cprovl:target>
123                     <cprovl:ID cprovl:var_type="cprovd:s-ref" cprovl:var_identifier="r:ag001"
                          ↪    cprovl:category_type="cprovd:Subject">r:ag001</cprovl:ID>
124             </cprovl:target>
125     </cprovl:IF>
126     <cprovl:THEN>
127             <cprovl:entity prov:id="confidenshare:result">
128                     <cprovl:actionType>cprovd:Deny</cprovl:actionType>
129                     <cprovl:resourceId>confidenshare:share</cprovl:resourceId>
130             </cprovl:entity>
131             <cprov:hadOwnership>
132                     <prov:entity prov:ref="confidenshare:result" />
133                     <prov:agent prov:ref="r:ag001" />
134                     <cprov:ownershipType>cprovd:possession</cprov:ownershipType>
135             </cprov:hadOwnership>
136     </cprovl:THEN>
137 </cprovl:RULE>
138 </cprovl:POLICY>
```

A.7.14 Traceability Graph for Policy 7

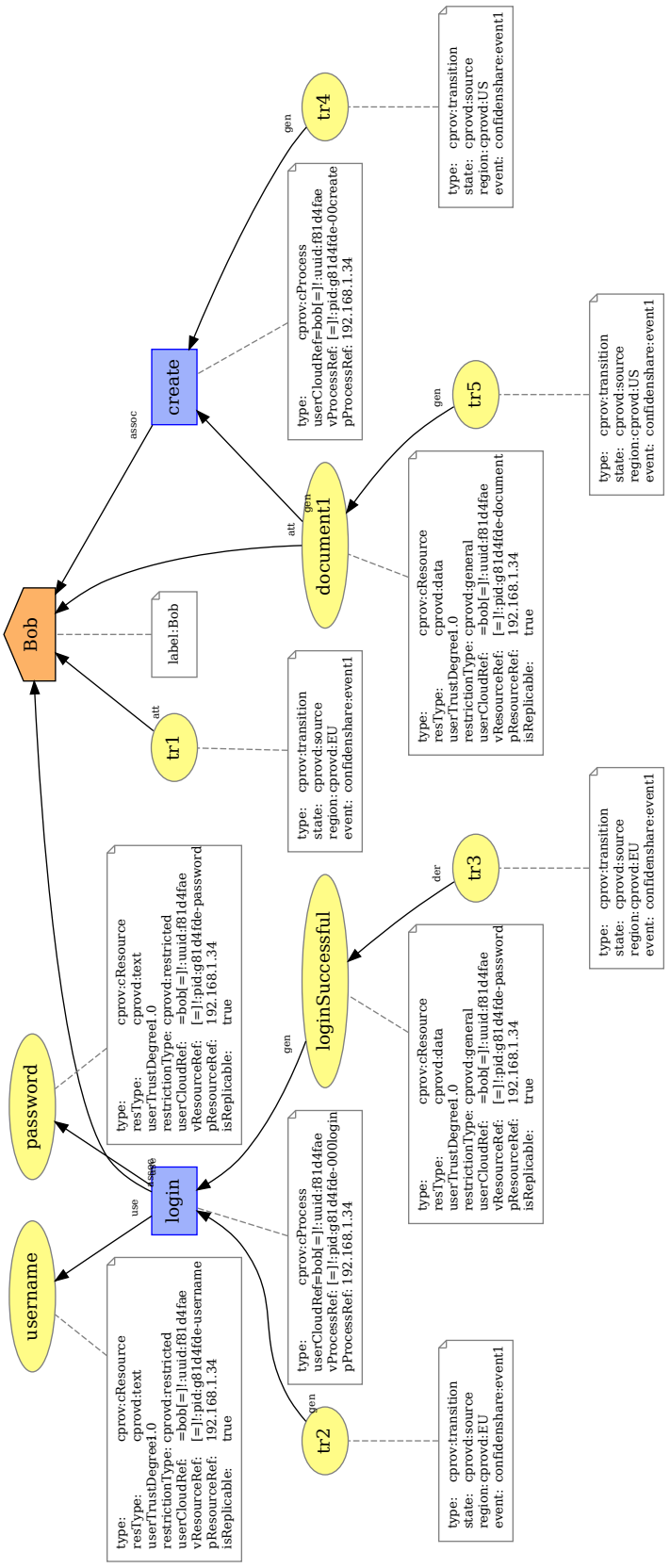


FIGURE A.7: Traceability Graph 7

A.7.15 policy8

```
<?xml version="1.0" encoding="UTF-8"?>
<cprovl:POLICY xmlns:cprov="http://labs.orange.com/uk/cprov#" xmlns:cprovl="http://labs.orange.com/uk/cprovl#"
        xmlns:prov="http://www.w3.org/ns/prov#" xmlns:cprovd="http://labs.orange.com/uk/cprovd#"
        xmlns:r="http://labs.orange.com/uk/r#" xmlns:cu="http://www.w3.org/1999/xhtml/datatypes/"
        xmlns:confidenshare="http://labs.orange.com/uk/confidenshare#" xmlns:opmx="http://openprovenance.org/model/opmx#"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://labs.orange.com/uk/cprovl#
        file:/Users/mufy/Dropbox/EngD/year3/schema/cProvl/cProvl-v1.1.xsd">

        <cprovl:entity prov:id="confidenshare:policy8">
                <cprovl:description>
                        A system cannot replicate a file, if the user explicitly marks during the creation
                        of a file as non-replicable
                </cprovl:description>
        </cprovl:entity>

        <cprovl:ENV>
                <cprovl:entity prov:id="confidenshare:rep.store.confidenshare.labs.orange.com">
                        <cprovl:description/>
                </cprovl:entity>
        </cprovl:ENV>

        <cprovl:RULE>
                <cprovl:entity prov:id="confidenshare:rule1">
                        <cprovl:description>
                        </cprovl:description>
                </cprovl:entity>
                <cprovl:DEF>
                        <cprovl:entity prov:id="confidenshare:def">
                                <cprovl:range>cprovd:All</cprovl:range>
                        </cprovl:entity>
                </cprovl:DEF>
                <cprovl:IF cprovl:quantifier="cprovd:There-exist">

                        <!-- Check for the request resource -->
                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:d-ref"
                                 ↪    cprovl:category_type="cprovd:Resource">r:e001</cprovl:ID>
                        </cprovl:target>
                        <!-- Check for the process replication -->
                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:d-ref"
                                 ↪    cprovl:category_type="cprovd:Action">confidenshare:replicate</cprovl:ID>
                        </cprovl:target>

                </cprovl:IF>
                <cprovl:SUCH-THAT>
                        <cprovl:statements>

                                <!-- Check if the resource is replicable -->
                                <cprovl:statement>
                                        <cprov:cResource prov:id="r:e001">
                                                <cprov:isReplicable>true</cprov:isReplicable>
                                        </cprov:cResource>
                                </cprovl:statement>

                        </cprovl:statements>
                </cprovl:SUCH-THAT>
                <cprovl:THEN>
                        <cprovl:entity prov:id="confidenshare:result">
                                <cprovl:actionType>cprovd:Permit</cprovl:actionType>
                                <cprovl:resourceId>confidenshare:replicate</cprovl:resourceId>
                        </cprovl:entity>
                        <cprov:hadOwnership>
                                <prov:entity prov:ref="confidenshare:result" />
                                <prov:agent prov:ref="r:ag001" />
                                <cprov:ownershipType>cprovd:possession</cprov:ownershipType>
                        </cprov:hadOwnership>
                </cprovl:THEN>
        </cprovl:RULE>

        <cprovl:RULE>
                <cprovl:entity>
                        <cprovl:description>Otherwise deny all</cprovl:description>
                </cprovl:entity>
                <cprovl:DEF>
                        <cprovl:entity>
```

```
75                              <cprovl:range>cprovd:All</cprovl:range>
76                      </cprovl:entity>
77              </cprovl:DEF>
78              <cprovl:IF cprovl:quantifier="cprovd:There-exist">
79                      <cprovl:target>
80                              <cprovl:ID cprovl:var_type="cprovd:s-ref" cprovl:var_identifier="r:ag001"
                                  ↪    cprovl:category_type="cprovd:Subject">r:ag001</cprovl:ID>
81                      </cprovl:target>
82              </cprovl:IF>
83              <cprovl:THEN>
84                      <cprovl:entity prov:id="confidenshare:result">
85                              <cprovl:actionType>cprovd:Deny</cprovl:actionType>
86                              <cprovl:resourceId>confidenshare:share</cprovl:resourceId>
87                      </cprovl:entity>
88                      <cprov:hadOwnership>
89                              <prov:entity prov:ref="confidenshare:result" />
90                              <prov:agent prov:ref="r:ag001" />
91                              <cprov:ownershipType>cprovd:possession</cprov:ownershipType>
92                      </cprov:hadOwnership>
93              </cprovl:THEN>
94      </cprovl:RULE>
95 </cprovl:POLICY>
```

A.7.16 Traceability Graph for Policy 8

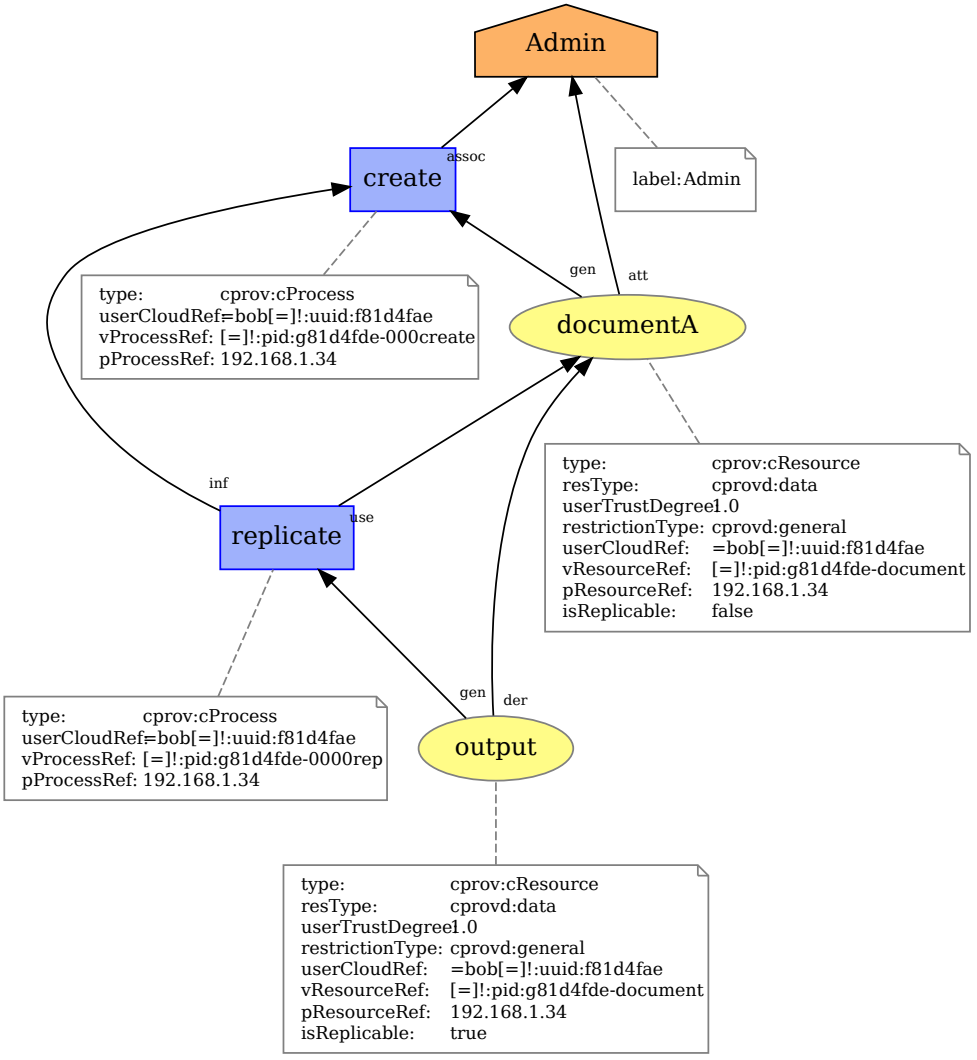


FIGURE A.8: Traceability Graph 6

A.7.17 policy9

```
<?xml version="1.0" encoding="UTF-8"?>
<cprovl:POLICY xmlns:cprov="http://labs.orange.com/uk/cprov#" xmlns:cprovl="http://labs.orange.com/uk/cprovl#"
        xmlns:prov="http://www.w3.org/ns/prov#" xmlns:cprovd="http://labs.orange.com/uk/cprovd#"
        xmlns:r="http://labs.orange.com/uk/r#" xmlns:cu="http://www.w3.org/1999/xhtml/datatypes/"
        xmlns:confidenshare="http://labs.orange.com/uk/confidenshare#" xmlns:opmx="http://openprovenance.org/model/opmx#"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://labs.orange.com/uk/cprovl#
        file:/Users/mufy/Dropbox/EngD/year3/schema/cProvl/cProvl-v1.1.xsd">

        <cprovl:entity prov:id="confidenshare:policy9">
                <cprovl:description>
                        A user (userA) can share a file, which is marked as non-modifiable, to another user
                        (userB), this user (userB) cannot share the file if it has been modified by him/herself
                        (userB).
                </cprovl:description>
        </cprovl:entity>

        <cprovl:ENV>
                <cprovl:entity prov:id="confidenshare:mod.share.confidenshare.labs.orange.com">
                        <cprovl:description/>
                </cprovl:entity>
        </cprovl:ENV>

        <cprovl:RULE>
                <cprovl:entity prov:id="confidenshare:rule1">
                        <cprovl:description>
                        </cprovl:description>
                </cprovl:entity>
                <cprovl:DEF>
                        <cprovl:entity prov:id="confidenshare:def">
                                <cprovl:range>cprovd:All</cprovl:range>
                        </cprovl:entity>
                </cprovl:DEF>
                <cprovl:IF cprovl:quantifier="cprovd:There-exist">

                        <!-- Check input request resource with the store -->
                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:d-ref" cprovl:var_identifier="r:fileA"
                                 ↪    cprovl:category_type="cprovd:Resource">r:fileA</cprovl:ID>
                        </cprovl:target>

                        <!-- Check for the request user -->
                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:d-ref" cprovl:var_identifier="r:ag001"
                                 ↪    cprovl:category_type="cprovd:Subject">r:ag001</cprovl:ID>
                        </cprovl:target>

                        <!-- Process for sharing -->
                        <cprovl:target>
                                <cprovl:ID cprovl:category_type="cprovd:Action">confidenshare:share</cprovl:ID>
                        </cprovl:target>

                        <!-- variable to populate new midification -->
                        <cprovl:target>
                                <cprovl:ID cprovl:var_type="cprovd:new"
                                 ↪    cprovl:var_identifier="r:newModification">r:newModification</cprovl:ID>
                        </cprovl:target>

                </cprovl:IF>
                <cprovl:SUCH-THAT>

                        <cprovl:statements>
                                <!-- Check if he/she is NOT the owner (owner does not need to have such restriction)
                                        -->
                                <cprovl:statement cprovl:negate="true">
                                        <cprov:hadOwnership prov:id="confidenshare:hd1">
                                                <prov:entity prov:ref="r:fileA" />
                                                <prov:agent prov:ref="r:ag001" />
                                                <cprov:ownershipType>cprovd:originator</cprov:ownershipType>
                                        </cprov:hadOwnership>
                                </cprovl:statement>

                                <!-- Check if the file was derived from another file-->
                                <cprovl:statement>
                                        <cprov:wasRepresentationOf prov:id="confidenshare:wro1">
                                                <prov:generatedEntity prov:ref="r:fileA" />
                                                <prov:usedEntity prov:ref="r:newModification" />
```

```
74                                      </cprov:wasRepresentationOf>
75                              </cprovl:statement>
76
77
78                              <!-- Check if ancestor file is 'non-modifiable' -->
79                              <cprovl:statement cprovl:negate="true">
80                                      <cprov:cResource prov:id="r:newModification">
81                                              <cprov:restrictionType>cprovd:non-modifiable</cprov:restrictionType>
82                                      </cprov:cResource>
83                              </cprovl:statement>
84
85                      </cprovl:statements>
86              </cprovl:SUCH-THAT>
87              <cprovl:THEN>
88                      <cprovl:entity prov:id="confidenshare:result">
89                              <cprovl:actionType>cprovd:Permit</cprovl:actionType>
90                              <cprovl:resourceId>confidenshare:share</cprovl:resourceId>
91                      </cprovl:entity>
92                      <cprov:hadOwnership>
93                              <prov:entity prov:ref="confidenshare:result" />
94                              <prov:agent prov:ref="confidenshare:ag001" />
95                              <cprov:ownershipType>cprovd:possession</cprov:ownershipType>
96                      </cprov:hadOwnership>
97              </cprovl:THEN>
98      </cprovl:RULE>
99
100     <cprovl:RULE>
101             <cprovl:entity>
102                     <cprovl:description>Otherwise deny all</cprovl:description>
103             </cprovl:entity>
104             <cprovl:DEF>
105                     <cprovl:entity>
106                             <cprovl:range>cprovd:All</cprovl:range>
107                     </cprovl:entity>
108             </cprovl:DEF>
109             <cprovl:IF cprovl:quantifier="cprovd:There-exist">
110                     <cprovl:target>
111                             <cprovl:ID cprovl:var_type="cprovd:s-ref" cprovl:var_identifier="r:ag001"
                                 ↪    cprovl:category_type="cprovd:Subject">r:ag001</cprovl:ID>
112                     </cprovl:target>
113             </cprovl:IF>
114             <cprovl:THEN>
115                     <cprovl:entity prov:id="confidenshare:result">
116                             <cprovl:actionType>cprovd:Deny</cprovl:actionType>
117                             <cprovl:resourceId>confidenshare:share</cprovl:resourceId>
118                     </cprovl:entity>
119                     <cprov:hadOwnership>
120                             <prov:entity prov:ref="confidenshare:result" />
121                             <prov:agent prov:ref="r:ag001" />
122                             <cprov:ownershipType>cprovd:possession</cprov:ownershipType>
123                     </cprov:hadOwnership>
124             </cprovl:THEN>
125     </cprovl:RULE>
126 </cprovl:POLICY>
```

A.7.18 Traceability Graph for Policy 9

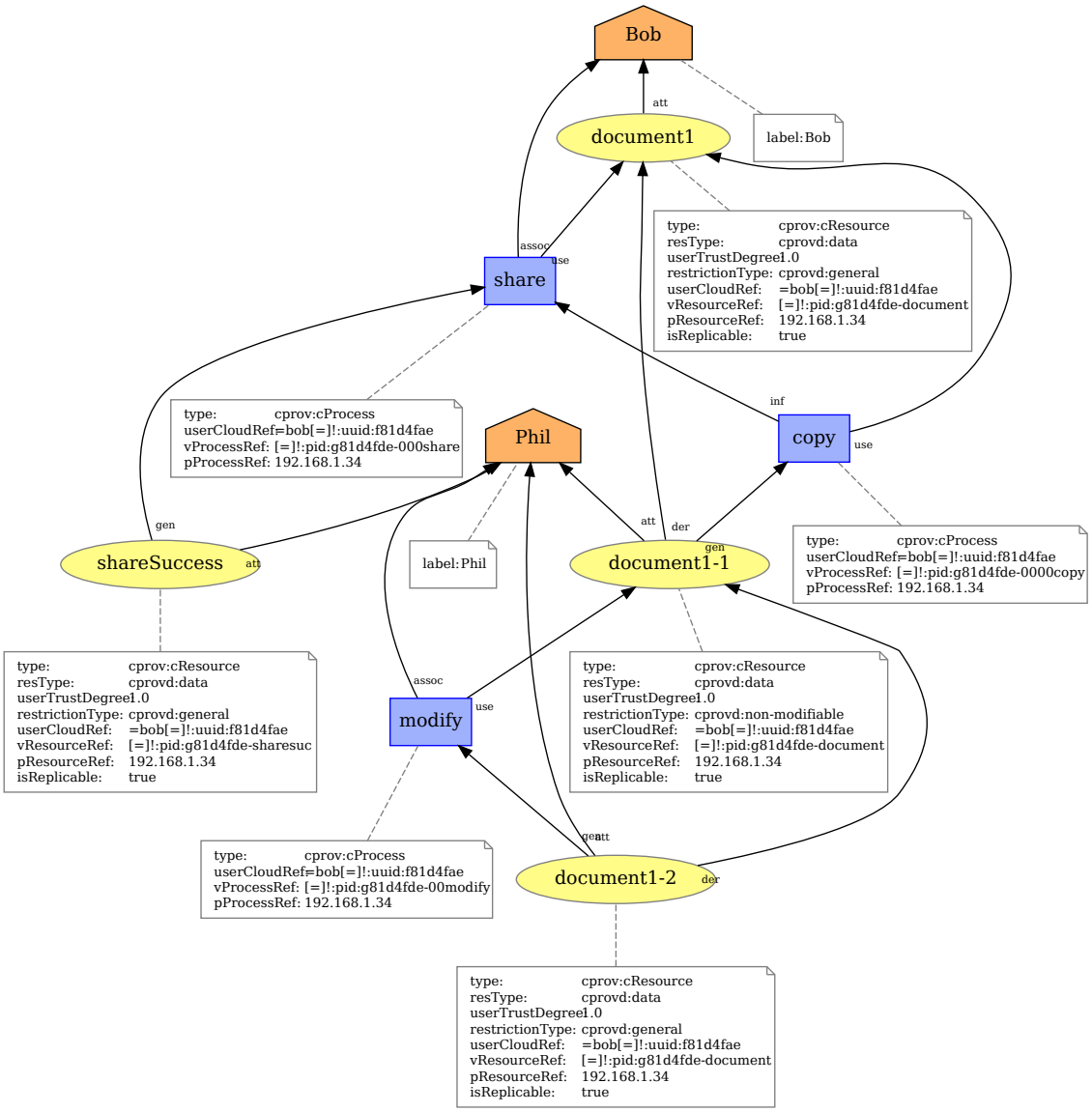


FIGURE A.9: Traceability Graph 9

A.8 cProv Traceability Model Client API

```java
/**
 * @file                ServiceTraceability.java
 * @project         CloudProvenance
 * @Module               provenanceModel
 * @date                18 05 2013
 * @version         1.0
 */
package com.provenance.cloudprovenance.service.traceability.api;

import java.util.Date;

/**
 * This interface defines methods for nodes and edges for traceability model
 *
 * @author lbum7009
 *
 */
public interface ServiceTraceability<T> {
        /**
         * T = defines type, it can be of XML, JSON or String
         */

        // Nodes
        public T pResource(String id, String resType, String ip, String MAC,
                        String hostType, Date timeStamp, String restrictionType, float trustValue);

        public T cResource(String id, String resType, String userCloudRef,
                        String vResourceRef, String pResourceRef, boolean isReplicable,
                        Date TTL, Date timeStamp, String restrictionType, float trustValue, String des);

        public T cProcess(String id, String userCloudRef, String vProcessRef,
                        String pProcessRef, Date timeStamp);

        public T transition(String id, String state, String country,
                        String latitude, String longitude, String region, String eventId);

        // Node from Prov

        public T Agent(String id, String name, String description);

        // Edges
        public T wasImplicitlyCalledBy(String id, String informed, String infromant,
                        String type, String callComm, String callMedium, String callNetwork);

        // TODO: Identify the differences between the both

        public T wasExplicitlyCalledBy(String id, String informed, String informant,
                        String type, String callComm, String callMedium, String callNetwork);

        public T wasRecurrentlyCalledBy(String id, String informed, String informand,
                        String type, String callComm, long timeout);

        public T hadOwnership(String id, String resource, String agent,
                        String ownershipType);

        // TODO:- What is generation, explore more
        public T wasRepresentationOf(String id, String generatedResource,
                        String usedResource, String processInvolved, String generation,
                        String usage, String method);

        public T wasReferenceOf(String id, String generatedResource,
                        String usedResource, String processInvolved, String generation,
                        String usage, String method, String type);

        public T wasVirtualizedBy(String id, String generatedResource,
                        String processInvolved, Date time, String purpose);

        public T hadTransitionState_A(String id, String generatedResource,
                        String transResource, String method);

        public T hadTransitionState_B(String id, String generatedActivity,
                        String transResource, String method);

        public T hadTransitionState_C(String id, String generatedAgent,
                        String stateResource, String method);
```

```
76
77          // edge from Prov
78
79          public T used(String id, String processInvolved, String generatedResource,
80                          Date date);
81
82          public T wasInitiallyCalledBy(String id, String processInvolved,
83                          String involvedAgent, String plan, String accessMedium,
84                          String accessNetwork);
85
86          //public T getCurrentTraceabilityDocument();
87
88
89  }
```

/** XML- based interface */

```
1  /**
2   * @file                XmlServiceTraceability.java
3   * @project         CloudProvenance
4   * @Module               provenanceModel
5   * @date                18 05 2013
6   * @version         1.0
7   */
8
9  package com.provenance.cloudprovenance.service.traceability.api;
10
11 import com.provenance.cloudprovenance.traceabilityModel.generated.TraceabilityDocument;
12
13 /**
14  * This interface defines specific methods related to XML based data traceability
15  *
16  * @author lbum7009
17  *
18  */
19 public interface ServiceXmlDocumentTraceability<T> extends ServiceTraceability<T> {
20
21          public TraceabilityDocument getRootNode();
22
23          public void InitilizeTraceabilityDocument();
24
25          public boolean isTraceabilityDocumentEmpty();
26
27          public void setCurrentTraceabilityDocument(T t);
28
29          public T getCurrentTraceabilityDocument();
30
31          public void setMaxStatementPerDocument(int max);
32
33 }
```

A.9 cProvl Policy Language Client API

```
1  /**
2   * ServiceComplianceAPI.java
3   *
4   */
5  package com.provenance.cloudprovenance.service.policy.api;
6
7  /**
8   * @author lbwm7009
9   *
10  */
11 public interface ServiceCompliance<T> {
12
13
14         /* construct a policy request with single user, resource and process */
15         public T constructRequest(String requestUserId, String resourceId, String processId, String environmentId);
16
17         /* construct a policy request with multiple users, resources and processes */
18         public T constructRequest(String requestUserIds[], String resourceIds[], String processIds[], String environmentId);
19
20
21         /* obtain a response */
22         public T getResponse();
23
24 }
```

Bibliography

[1] Regulation (ec) no 178/2002 of the european parliament and of the council of 28 january2002 laying down the general principles and requirements of food law, establishing the european food safetyauthorityand laying down procedures in matters of food safety. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2002:031:0001:0024:EN:PDF, 2002.

[2] Food traceability. http://ec.europa.eu/food/food/foodlaw/traceability/factsheet_trace_2007_en.pdf, 2007.

[3] Amazon elastic compute cloud (amazon ec2). http://aws.amazon.com/ec2/, 2011.

[4] I.M. Abbadi and J. Lyle. Challenges for provenance in cloud computing. In *TaPP*, 2011. URL https://www.usenix.org/legacy/events/tapp11/tech/final_files/Abbadi.pdf.

[5] D. Abi Haidar, N. Cuppens-Boulahia, F. Cuppens, and H. Debar. An extended rbac profile of xacml. In *Proceedings of the 3rd ACM workshop on Secure web services*, pages 13–22. ACM, 2006. doi:10.1145/1180367.1180372.

[6] K.Z. Ahmed and C.E. Umrysh. *Developing enterprise Java applications with J2EE and UML*. Addison-Wesley, 2002.

[7] R. Aldeco-Perez and L. Moreau. Provenance-based auditing of private data use. In *Proc. of the BCS International Academic Research Conference, Visions of Computer Science*. Citeseer, 2008. URL http://eprints.soton.ac.uk/266580/.

[8] R. Aldeco-Prez and L. Moreau. A provenance-based compliance framework. In ArneJ. Berre, Asuncin Gmez-Prez, Kurt Tutschku, and Dieter Fensel, editors, *Future Internet - FIS 2010*, volume 6369 of *Lecture Notes in Computer Science*, pages 128–137. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15876-6. doi:10.1007/978-3-642-15877-3_14. URL http://dx.doi.org/10.1007/978-3-642-15877-3_14.

[9] M. Ali. Green cloud on the horizon. In MartinGilje Jaatun, Gansen Zhao, and Chunming Rong, editors, *Cloud Computing*, volume 5931 of *Lecture Notes in Com-*

puter Science, pages 451–459. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-10664-4. doi:10.1007/978-3-642-10665-1_41. URL http://dx.doi.org/10.1007/978-3-642-10665-1_41.

[10] M. Ali. Can a mobile cloud be more trustworthy than a traditional cloud? In Ramjee Prasad, Kroly Farkas, AndreasU. Schmidt, Antonio Lioy, Giovanni Russello, and FlaminiaL. Luccio, editors, *Security and Privacy in Mobile Information and Communication Systems*, volume 94 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 125–135. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-30243-5. doi:10.1007/978-3-642-30244-2_11. URL http://dx.doi.org/10.1007/978-3-642-30244-2_11.

[11] M.L Ambrose and J. Ausloos. The right to be forgotten across the pond. *Journal of Information Policy*, 3:1–23, 2013. ISSN 21583897. URL http://www.jstor.org/stable/10.5325/jinfopoli.3.2013.0001.

[12] K. Anand, S. Bowers, T.M. McPhillips, and B. Ludaescher. Exploring scientific workflow provenance using hybrid queries over nested data and lineage graphs. In *Proceedings of 21st International Conference on Scientific and Statistical Database Management (SSDBM'09)*, pages 237–254, New Orleans, LA, USA, 2009. doi:http://dx.doi.org/10.1007/978-3-642-02279-1_18. URL http://daks.ucdavis.edu/~sbowers/ssdbm-09.pdf.

[13] A. Anderson. Xacml profile for role based access control (rbac). *OASIS Access Control TC committee draft*, 1:13, 2004. URL http://xml.coverpages.org/OASIS-XACML-RBACProfile.pdf.

[14] A.H. Anderson. An introduction to the web services policy language (wspl). In *Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on*, pages 189–192, june 2004. doi:10.1109/POLICY.2004.1309166.

[15] P. Anderson and J. Cheney. Toward provenance-based security for configuration languages. In *Presented as part of the 4th USENIX Workshop on the Theory and Practice of Provenance*, Boston, MA, 2012. USENIX. URL https://www.usenix.org/conference/tapp12/workshop-program/presentation/Anderson.

[16] S. Andreozzi, S. Burke, F. Ehm, L. Field, G. Galang, B. Konya, M. Litmaath, M. Litmaath, and JP. Navarro. Glue specification v. 2.0. Technical report, Open Grid Forum, 2009. URL http://lup.lub.lu.se/record/1454660.

[17] S.J. Andriole. Seven indisputable technology trends that will define 2015. *Communications of the Association for Information Systems*, 30(1):4, 2012. URL http://aisel.aisnet.org/cais/vol30/iss1/4.

[18] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al. A view of cloud computing. *Commun. ACM*, 53(4):50–58, April 2010. ISSN 0001-0782. doi:10.1145/1721654.1721672. URL http://doi.acm.org/10.1145/1721654.1721672.

[19] J. Arthur and S. Azadegan. Spring framework for rapid open source j2ee web application development: A case study. In *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 2005 and First ACIS International Workshop on Self-Assembling Wireless Networks. SNPD/SAWN 2005. Sixth International Conference on*, pages 90–95. IEEE, 2005. doi:10.1109/SNPDSAWN.2005.74.

[20] G. Ateniese, K. Benson, and S. Hohenberger. Key-private proxy re-encryption. In Marc Fischlin, editor, *Topics in Cryptology CT-RSA 2009*, volume 5473 of *Lecture Notes in Computer Science*, pages 279–294. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-00861-0. doi:10.1007/978-3-642-00862-7_19. URL http://dx.doi.org/10.1007/978-3-642-00862-7_19.

[21] M. Bailey, D. Bibby, R. Boggs, J. Bozman, B. Burke, C. Christiansen, L. DuBois, M. Eastwood, M. Fauscette, J. Gantz, et al. It cloud services forecast ? 2008, 2012: A key driver of new growth. http://blogs.idc.com/ie/?p=224, 2008.

[22] S. Bajaj, D. Box, D. Chappell, F. Curbera, G. Daniels, P. Hallam-Baker, M. Hondo, C. Kaler, D. Langworthy, A. Nadalin, et al. Web services policy 1.2-framework (ws-policy). *W3C Member Submission*, 25:12, 2006. URL http://firebatcon.googlecode.com/svn-history/r336/trunk/firebatCon/books/webservice/WS_POLICY.pdf.

[23] T. Baker. A grammar of dublin core. *D-lib magazine*, 6(10):47–60, 2000.

[24] Z. Bao, S. Cohen-Boulakia, S.B. Davidson, A. Eyal, and S. Khanna. Differencing provenance in scientific workflows. In *IEEE 25th International Conference on Data Engineering (ICDE'09)*, pages 808–819. IEEE Computer Society, 2009. doi:http://doi.ieeecomputersociety.org/10.1109/ICDE.2009.103. URL http://www.cis.upenn.edu/~zhuowei/docs/PDiffView/ICDEResearchLong-ZBao-diff.pdf.

[25] R.S. Barga and L.A. Digiampietri. Automatic generation of workflow provenance. In Moreau L. and Foster I., editors, *Proceedings of the International Provenance and Annotation Workshop 2006 (IPAW'2006)*, volume 4145, pages 1–9. Springer, 2006. doi:http://dx.doi.org/10.1007/11890850_1. URL http://www.springerlink.com/content/1585u07547832814/?p=1eb5e64187c647a0931eecb0993154ed&pi=0.

[26] P.R. Barham, B. Dragovic, K.A. Fraser, S.M. Hand, T.L. Harris, A.C. Ho, E. Kotsovinos, AV Madhavapeddy, R. Neugebauer, I.A. Pratt, et al. Xen 2002.

University of Cambridge, Computer Laboratory, Tech. Rep. UCAM-CL-TR-553, Jan, 2003. URL http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-553.pdf.

[27] S. Barker and P.J. Stuckey. Flexible access control policy specification with constraint logic programming. *ACM Trans. Inf. Syst. Secur.*, 6(4):501–546, November 2003. ISSN 1094-9224. doi:10.1145/950191.950194. URL http://doi.acm.org/10.1145/950191.950194.

[28] S.A. Baset. Open source cloud technologies. In *Proceedings of the Third ACM Symposium on Cloud Computing*, SoCC '12, pages 28:1–28:2, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1761-0. doi:10.1145/2391229.2391257. URL http://doi.acm.org/10.1145/2391229.2391257.

[29] A. Bates, B. Mood, M. Valafar, and K. Butler. Towards secure provenance-based access control in cloud environments. In *Proceedings of the Third ACM Conference on Data and Application Security and Privacy*, CODASPY '13, pages 277–284, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1890-7. doi:10.1145/2435349.2435389. URL http://doi.acm.org/10.1145/2435349.2435389.

[30] S. Bechhofer. Owl: Web ontology language. In L. LIU and M.T. ZSU, editors, *Encyclopedia of Database Systems*, pages 2008–2009. Springer US, 2009. ISBN 978-0-387-35544-3. doi:10.1007/978-0-387-39940-9_1073. URL http://dx.doi.org/10.1007/978-0-387-39940-9_1073.

[31] K. Beck and C. Andres. *Extreme programming explained: embrace change*. Addison-Wesley Professional, 2004.

[32] R. Bellis. Dns proxy implementation guidelines. 2009. URL http://tools.ietf.org/html/rfc5625.

[33] G.J. Bex, F. Neven, and J. Van den Bussche. Dtds versus xml schema: A practical study. In *Proceedings of the 7th International Workshop on the Web and Databases: Colocated with ACM SIGMOD/PODS 2004*, WebDB '04, pages 79–84, New York, NY, USA, 2004. ACM. doi:10.1145/1017074.1017095. URL http://doi.acm.org/10.1145/1017074.1017095.

[34] O. Biton, S. Cohen-Boulakia, S.B. Davidson, and C.S. Hara. Querying and managing provenance through user views in scientific workflows. In *International Conference Data Engineering (ICDE'08)*, pages 1072–1081, Los Alamitos, CA, USA, 2008. IEEE Computer Society. ISBN 978-1-4244-1836-7. doi:http://doi.ieeecomputersociety.org/10.1109/ICDE.2008.4497516. URL http://www.inf.ufpr.br/carmem/pub/icde08.pdf.

[35] S. Bowers, T.M. McPhillips, and B. Ludäscher. Provenance in collection-oriented scientific workflows. *Concurrency and Computation: Practice and Experience*, 20 (5):519–529, 2008.

[36] T. Bray, J. Paoli, C.M Sperberg-McQueen, E. Maler, and F. Yergeau. Extensible markup language (xml). *World Wide Web Consortium Recommendation REC-xml-19980210*, 1998. URL http://www.w3.org/TR/1998/REC-xml-19980210.

[37] J.G. Breslin, S. Decker, A. Harth, and U. Bojars. Sioc: an approach to connect web-based communities. *International Journal of Web Based Communities*, 2(2): 133–142, 2006. doi:10.1504/IJWBC.2006.010305.

[38] D. Brickley and L. Miller. Foaf vocabulary specification 0.98. *Namespace document*, 9, 2012. URL http://ontogenealogy.com/documents/2012/08/foaf-vocabulary-specification-0-98-20100809.pdf.

[39] S. Buckingham Shum, R. Slack, M. Daw, B. Juby, A. Rowley, M. Bachler, C. Mancini, D. Michaelides, R. Procter, D. de Roure, et al. Memetic: an infrastructure for meeting memory. In *Proceeding of the 2006 conference on Cooperative Systems Design: Seamless Integration of Artifacts and Conversations–Enhanced Concepts of Infrastructure for Communication*, pages 71–85. IOS Press, 2006. URL http://kmi.open.ac.uk/publications/pdf/KMI-TR-06-02.pdf.

[40] P. Buneman, S. Khanna, and T. Wang-Chiew. Why and where: A characterization of data provenance. In Jan Van den Bussche and Victor Vianu, editors, *Database Theory ICDT 2001*, volume 1973 of *Lecture Notes in Computer Science*, pages 316–330. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-41456-8. doi:10.1007/3-540-44503-X_20. URL http://dx.doi.org/10.1007/3-540-44503-X_20.

[41] M. Burstein, J. Hobbs, O. Lassila, D. Mcdermott, S. Mcilraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara. OWL-S: Semantic Markup for Web Services. Website, November 2004. URL http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/.

[42] T. Cadenhead, V. Khadilkar, M. Kantarcioglu, and B. Thuraisingham. A language for provenance access control. In *Proceedings of the First ACM Conference on Data and Application Security and Privacy*, CODASPY '11, pages 133–144, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0466-5. doi:10.1145/1943513.1943532. URL http://doi.acm.org/10.1145/1943513.1943532.

[43] T. Cadenhead, V. Khadilkar, M. Kantarcioglu, and B. Thuraisingham. Transforming provenance using redaction. In *Proceedings of the 16th ACM Symposium on Access Control Models and Technologies*, SACMAT '11, pages 93–102, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0688-1. doi:10.1145/1998441.1998456. URL http://doi.acm.org/10.1145/1998441.1998456.

[44] J.J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named graphs, provenance and trust. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 613–622, New York, NY, USA, 2005. ACM. ISBN 1-59593-046-9. doi:10.1145/1060745.1060835. URL http://doi.acm.org/10.1145/1060745.1060835.

[45] D. Ceolin, P.T Groth, W.R. Van Hage, A. Nottamkandath, and W. Fokkink. Trust evaluation through user reputation and provenance analysis. *URSW*, 900:15–26, 2012. URL http://ceur-ws.org/Vol-900/proceedings.pdf#page=23.

[46] V.G Cerf. Ascii format for network interchange. 1969. URL http://tools.ietf.org/html/rfc20.html.

[47] L. Chen, V. Tan, F. Xu, A. Biller, P. Groth, S. Miles, J. Ibbotson, M. Luck, and L. Moreau. A proof of concept: Provenance in a service oriented architecture. In *Proceedings of the Fourth All Hands Meeting (AHM)*. Citeseer, 2005. URL http://www.allhands.org.uk/2005/proceedings/papers/503.pdf.

[48] L. Chen, L. Gasparini, and T.J Norman. Xacml and risk-aware access control. *Resource*, 2(10):3–5, 2013. URL http://homepages.abdn.ac.uk/t.j.norman/pages/pdfs/Chen++WOSIS2013.pdf.

[49] J. Cheney. A formal framework for provenance security. In *Computer Security Foundations Symposium (CSF), 2011 IEEE 24th*, pages 281–293, June 2011. doi:10.1109/CSF.2011.26.

[50] J. Cheney, L. Chiticariu, and W.C Tan. *Provenance in databases: Why, how, and where*, volume 4. Now Publishers Inc, 2009.

[51] L. Childers, T. Disz, R. Olson, M.E. Papka, R. Stevens, and T. Udeshi. Access grid: Immersive group-to-group collaborative visualization. In *Proc. 4th International Immersive Projection Technology Workshop*. Citeseer, 2000. URL http://www-unix.mcs.anl.gov/fl/publications/childers00.pdf.

[52] S.S.M. Chow, C.K. Chu, X. Huang, J. Zhou, and R.H. Deng. Dynamic secure cloud storage with provenance. In David Naccache, editor, *Cryptography and Security: From Theory to Applications*, volume 6805 of *Lecture Notes in Computer Science*, pages 442–464. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-28367-3. doi:10.1007/978-3-642-28368-0_28. URL http://dx.doi.org/10.1007/978-3-642-28368-0_28.

[53] P. Ciccarese, D. Shotton, S. Peroni, and T. Clark. Cito+ swan: The web semantics of bibliographic records, citations, evidence and discourse relationships. *Semantic Web*, 5(4):295–311, 2014. URL http://speroni.web.cs.unibo.it/publications/ciccarese-2014-cito-swan-semantics.pdf.

[54] E. Ciurana. Google app engine. In *Developing with Google App Engine*, pages 1–10. Apress, 2009. ISBN 978-1-4302-1831-9. doi:10.1007/978-1-4302-1832-6_1. URL http://dx.doi.org/10.1007/978-1-4302-1832-6_1.

[55] Apache CloudStack. Apache cloudstack open source cloud computing, 2014. URL https://cloudstack.apache.org/.

[56] D. Connolly. Naming and addressing: Uris, urls. *W3C Architecture Document*, 2002. URL http://www.w3.org/Addressing/.

[57] Unicode Consortium et al. *The Unicode Standard, Version 2.0*. Addison-Wesley Longman Publishing Co., Inc., 1997.

[58] V. Cuevas-Vicenttín, P. Kianmajd, B. Ludäscher, P. Missier, F. Chirigati, Y. Wei, D. Koop, and S. Dey. The pbase scientific workflow provenance repository. *International Journal of Digital Curation*, 9(2):28–38, 2014. doi:10.2218/ijdc.v9i2.332.

[59] P.P. Da Silva, D.L. McGuinness, and R. Fikes. A proof markup language for semantic web services. *Information Systems*, 31(45):381–395, 2006. ISSN 0306-4379. doi:http://dx.doi.org/10.1016/j.is.2005.02.003. URL http://www.sciencedirect.com/science/article/pii/S0306437905000281. The Semantic Web and Web Services.

[60] N. Dang, P. Jaehong, and R. Sandhu. Integrated provenance data for access control in group-centric collaboration. In *Information Reuse and Integration (IRI), 2012 IEEE 13th International Conference on*, pages 255–262, Aug 2012. doi:10.1109/IRI.2012.6303018.

[61] R. Danger, R.C Joy, J. Darlington, and V. Curcin. Access control for opm provenance graphs. In Paul Groth and James Frew, editors, *Provenance and Annotation of Data and Processes*, volume 7525 of *Lecture Notes in Computer Science*, pages 233–235. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-34221-9. doi:10.1007/978-3-642-34222-6_23. URL http://dx.doi.org/10.1007/978-3-642-34222-6_23.

[62] C.J. Date, C.J. White, and CJ Date. *A guide to DB2*, volume 3. Addison-Wesley New York, 1988.

[63] Y. Deng, M. Head, A. Kochut, J. Munson, A. Sailer, and H. Shaikh. An ontology based approach for cloud services catalog management. In PaulP. Maglio, Mathias Weske, Jian Yang, and Marcelo Fantinato, editors, *Service-Oriented Computing*, volume 6470 of *Lecture Notes in Computer Science*, pages 680–681. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-17357-8. doi:10.1007/978-3-642-17358-5_56. URL http://dx.doi.org/10.1007/978-3-642-17358-5_56.

[64] M. Dietz, S. Shekhar, Y. Pisetsky, A. Shu, and D.S. Wallach. Quire: Lightweight provenance for smart phone operating systems. In *20th USENIX Security*

Symposium, 2011. URL http://static.usenix.org/event/sec11/tech/full_papers/Dietz.pdf.

[65] Y. Doganata, K. Grueneberg, J. Karat, and N. Mukhi. Authoring and deploying business policies dynamically for compliance monitoring. In *Policies for Distributed Systems and Networks (POLICY), 2011 IEEE International Symposium on*, pages 161–164. IEEE, June 2011. doi:10.1109/POLICY.2011.27.

[66] E.J. Domingo, J.T. Nino, A.L. Lemos, M.L. Lemos, R.C. Palacios, and J.M.G. Berbis. Cloudio: A cloud computing-oriented multi-tenant architecture for business information systems. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 532–533, July 2010. doi:10.1109/CLOUD.2010.88.

[67] H.N. Duc. On a dilemma of conditional obligation. In *Analyomen*, volume 2, pages 93–100, 1997.

[68] S. Edward. Simpledb: a simple java-based multiuser syst for teaching database internals. In *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, SIGCSE '07, pages 561–565, New York, NY, USA, 2007. ACM. ISBN 1-59593-361-1. doi:10.1145/1227310.1227498. URL http://doi.acm.org/10.1145/1227310.1227498.

[69] O. Elkeelany, M.M. Matalgah, K.P. Sheikh, M. Thaker, G. Chaudhry, D. Medhi, and J. Qaddour. Performance analysis of ipsec protocol: encryption and authentication. In *Communications, 2002. ICC 2002. IEEE International Conference on*, volume 2, pages 1164–1168 vol.2. IEEE, 2002. doi:10.1109/ICC.2002.997033.

[70] W. Enck, P. Gilbert, B.G. Chun, L.P. Cox, J. Jung, P. McDaniel, and A.N. Sheth. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Trans. Comput. Syst.*, 32(2):5:1–5:29, June 2014. ISSN 0734-2071. doi:10.1145/2619091. URL http://doi.acm.org/10.1145/2619091.

[71] T. Erl. *Service-oriented architecture (SOA): concepts, technology, and design*. Prentice Hall, 2005. URL http://dspace.siu.ac.th/handle/1532/177.

[72] X. Feng, J. Shen, and Y. Fan. Rest: An alternative to rpc for web services architecture. In *Future Information Networks, 2009. ICFIN 2009. First International Conference on*, pages 7–10. IEEE, Oct 2009. doi:10.1109/ICFIN.2009.5339611.

[73] D. Ferraiolo, J. Cugini, and D.R. Kuhn. Role-based access control (rbac): Features and motivations. In *Proceedings of 11th Annual Computer Security Application Conference*, pages 241–48. NIST, 1995. URL http://csrc.nist.gov/groups/SNS/rbac/documents/ferraiolo-cugini-kuhn-95.pdf.

[74] D.F Ferraiolo, R. Sandhu, S. Gavrila, D.R Kuhn, and R. Chandramouli. Proposed nist standard for role-based access control. *ACM Trans. Inf. Syst. Secur.*, 4(3):

224–274, August 2001. ISSN 1094-9224. doi:10.1145/501978.501980. URL http://doi.acm.org/10.1145/501978.501980.

[75] J. Fialli and S. Vajjhala. The java architecture for xml binding (jaxb). *JSR Specification*, 2003. URL https://svn.java.net/svn/hj3~svn/trunk/etc/jaxb-1_0-fr-spec.pdf.

[76] F.T. Fonseca, M.J. Egenhofer, C.A. DavisJr, and K.A.V. Borges. Ontologies and knowledge sharing in urban {GIS}. *Computers, Environment and Urban Systems*, 24(3):251–272, 2000. ISSN 0198-9715. doi:http://dx.doi.org/10.1016/S0198-9715(00)00004-1. URL http://www.sciencedirect.com/science/article/pii/S0198971500000041.

[77] I. Foster, Z. Yong, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008*, pages 1–10, nov. 2008. doi:10.1109/GCE.2008.4738445.

[78] C. Fürber and M. Hepp. Using sparql and spin for data quality management on the semantic web. In Witold Abramowicz and Robert Tolksdorf, editors, *Business Information Systems*, volume 47 of *Lecture Notes in Business Information Processing*, pages 35–46. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-12813-4. doi:10.1007/978-3-642-12814-1_4. URL http://dx.doi.org/10.1007/978-3-642-12814-1_4.

[79] D. Garijo Verdejo, Ó. Corcho, and A. Gómez-Perez. Provenance and trust. In *Proceedings of the Doctoral Consortium of the 3rd Future Internet Symposium, DC-FIS 2010*, volume 623, Bonn, Alemania, 2010. CEUR-WS. URL http://oa.upm.es/6909/.

[80] J.M Garrido. Inter-process communication. In *Performance Modeling of Operating Systems Using Object-Oriented Simulation*, Series in Computer Science, pages 169–189. Springer US, 2000. ISBN 978-0-306-46459-1. doi:10.1007/0-306-46976-6_9. URL http://dx.doi.org/10.1007/0-306-46976-6_9.

[81] B. Glavic. Big data provenance: Challenges and implications for benchmarking. In Tilmann Rabl, Meikel Poess, Chaitanya Baru, and Hans-Arno Jacobsen, editors, *Specifying Big Data Benchmarks*, volume 8163 of *Lecture Notes in Computer Science*, pages 72–80. Springer Berlin Heidelberg, 2014. ISBN 978-3-642-53973-2. doi:10.1007/978-3-642-53974-9_7. URL http://dx.doi.org/10.1007/978-3-642-53974-9_7.

[82] S. Godik, A. Anderson, B. Parducci, P. Humenn, and S. Vajjhala. Oasis extensible access control 2 markup language (xacml) 3. Technical report, Tech. rep., OASIS, 2002. URL https://www.oasis-open.org/committees/xacml/repository/draft-xacml-schema-policy-13.pdf.

[83] J. Gosling. *The Java language specification*. Addison-Wesley Professional, 2000.

[84] S.J. Greenspan and C.L. McGowan. Structuring software development for reliability. *Microelectronics Reliability*, 17(1):75–83, 1978. doi:10.1016/0026-2714(78)91140-X.

[85] P. Groth and L Moreau. Prov-overview: an overview of the prov family of documents. 2013. URL http://www.w3.org/TR/2013/NOTE-prov-overview-20130430.

[86] P. Groth, S. Jiang, S. Miles, S. Munroe, V. Tan, S. Tsasakou, and L. Moreau. An architecture for provenance systems. Technical report, University of Southampton, February 2006. URL http://eprints.soton.ac.uk/263196/.

[87] T.R. Gruber et al. Toward principles for the design of ontologies used for knowledge sharing. *International journal of human computer studies*, 43(5):907–928, 1995. doi:10.1006/ijhc.1995.1081.

[88] K. Hamadache and P. Zerva. Provenance of feedback in cloud services. In *Service Oriented System Engineering (SOSE), 2014 IEEE 8th International Symposium on*, pages 23–34, April 2014. doi:10.1109/SOSE.2014.10.

[89] T. Han and K.M. Sim. An ontology-enhanced cloud service discovery system. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, pages 17–19, 2010. URL www.iaeng.org/publication/IMECS2010/IMECS2010_pp644-649.pdf.

[90] O. Hartig. Provenance information in the web of data. *LDOW*, 538, 2009. URL http://www.dbis.informatik.hu-berlin.de/fileadmin/research/papers/conferences/2009-ldow-hartig.pdf.

[91] R. Henjes, D. Schlosser, M. Menth, and V. Himmler. Throughput performance of the activemq jms server. In Torsten Braun, Georg Carle, and Burkhard Stiller, editors, *Kommunikation in Verteilten Systemen (KiVS)*, Informatik aktuell, pages 113–124. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-69961-3. doi:10.1007/978-3-540-69962-0_10. URL http://dx.doi.org/10.1007/978-3-540-69962-0_10.

[92] CN. Höfer and G. Karagiannis. Cloud computing services: taxonomy and comparison. *Journal of Internet Services and Applications*, 2(2):81–94, 2011. ISSN 1867-4828. doi:10.1007/s13174-011-0027-x. URL http://dx.doi.org/10.1007/s13174-011-0027-x.

[93] H. Hua, C. Tilmes, and S. Zednik. Prov-xml: The prov xml schema. Technical report, World Wide Web Consortium, April 2013. URL http://eprints.soton.ac.uk/356855/.

[94] T. Hughes and C. Martin. Design traceability of complex systems. In *Human Interaction with Complex Systems, 1998. Proceedings., Fourth Annual Symposium on*, pages 37–41, mar 1998. doi:10.1109/HUICS.1998.659952.

[95] T.D. Huynh and L. Moreau. Provstore: A public provenance repository. In Bertram Ludscher and Beth Plale, editors, *Provenance and Annotation of Data and Processes*, volume 8628 of *Lecture Notes in Computer Science*, pages 275–277. Springer International Publishing, 2015. ISBN 978-3-319-16461-8. doi:10.1007/978-3-319-16462-5_32. URL http://dx.doi.org/10.1007/978-3-319-16462-5_32.

[96] T.D. Huynh, P. Groth, and S. Zednik. Prov implementation report. Technical report, April 2013. URL http://eprints.soton.ac.uk/358440/.

[97] J. Ibbotson. Provenance and compliance. Technical report, IBM UK, 2006. URL http://www.gridprovenance.org/publications/ProvenanceCompliance.pdf.

[98] J. Ibbotson, D. Braines, D. Mott, S. Arunkumar, and M. Srivatsa. Documenting provenance with a controlled natural language. In *Annual Conference of the International Technology Alliance (ACITA)*, 2012. URL https://usukitacs.com/sites/default/files/P3_jibbotson_doc_provenance_cnl.pdf.

[99] M. Imran and H. Hlavacs. Applications of provenance data for cloud infrastructure. In *Semantics, Knowledge and Grids (SKG), 2012 Eighth International Conference on*, pages 16–23, Oct 2012. doi:10.1109/SKG.2012.21.

[100] R. Johnson, J. Hoeller, K. Donald, C. Sampaleanu, R. Harrop, T. Risberg, A. Arendsen, D. Davison, D. Kopylenko, M. Pollack, et al. The spring framework, reference documentation.", 2004.

[101] A.A.E. Kalam, R.E. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miege, C. Saurel, and G. Trouessin. Organization based access control. In *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*, pages 120–131, june 2003. doi:10.1109/POLICY.2003.1206966.

[102] J. Kandampully. Service quality to service loyalty: A relationship which goes beyond customer services. *Total Quality Management*, 9(6):431–443, 1998. doi:10.1080/0954412988370. URL http://dx.doi.org/10.1080/0954412988370.

[103] A. Keith and A. Ole. A comparison of software and hardware techniques for x86 virtualization. In *Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, ASPLOS XII, pages 2–13, New York, NY, USA, 2006. ACM. ISBN 1-59593-451-0. doi:10.1145/1168857.1168860. URL http://doi.acm.org/10.1145/1168857.1168860.

[104] A. Khalid. Cloud computing: Applying issues in small business. In *Signal Acquisition and Processing, 2010. ICSAP '10. International Conference on*, pages 278–281, Feb 2010. doi:10.1109/ICSAP.2010.78.

[105] K.M. Khan and Q. Malluhi. Establishing trust in cloud computing. *IT Professional*, 12(5):20–27, Sept 2010. ISSN 1520-9202. doi:10.1109/MITP.2010.128.

[106] C. Knobler. Digital evidence: The future of computing in law enforcement. 2009. URL http://www.csc.villanova.edu/~tway/courses/csc3990/f2009/csrs2009/Cory_Knobler_Digital_Forensics_CSRS_2009.pdf.

[107] D.E. Knuth. Backus normal form vs. backus naur form. *Commun. ACM*, 7(12): 735–736, December 1964. ISSN 0001-0782. doi:10.1145/355588.365140. URL http://doi.acm.org/10.1145/355588.365140.

[108] D. Kobialka. Ihs: Cloud spending will top $235b by 2017. http://talkincloud.com/cloud-computing-funding-and-finance/040714/ihs-cloud-spending-will-top-235b-2017, 2014.

[109] V. Kolovski, B. Parsia, Y. Katz, and J. Hendler. Representing web service policies in owl-dl. In Yolanda Gil, Enrico Motta, V.Richard Benjamins, and MarkA. Musen, editors, *The Semantic Web ISWC 2005*, volume 3729 of *Lecture Notes in Computer Science*, pages 461–475. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-29754-3. doi:10.1007/11574620_34. URL http://dx.doi.org/10.1007/11574620_34.

[110] R. Kumar, K. Jain, H. Maharwal, N. Jain, and A. Dadhich. Apache cloudstack: Open source infrastructure as a service cloud computing platform. In *International Journal of advancement in Engineering technology, Management and Applied Science*, 2014. URL http://ijrcemas.com/wp-content/uploads/2014/08/IJB024.pdf.

[111] P. Kumbhar and S. Krishnan. Sound data compression using different methods. In P.V. Krishna, M.R. Babu, and E. Ariwa, editors, *Global Trends in Computing and Communication Systems*, volume 269 of *Communications in Computer and Information Science*, pages 100–108. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-29218-7. doi:10.1007/978-3-642-29219-4_12. URL http://dx.doi.org/10.1007/978-3-642-29219-4_12.

[112] S. Lampoudi. The path to virtual machine images as first class provenance. *Age*, 2011. URL http://nidhogg.cs.ucsb.edu/research/tech_reports/reports/2012-05.pdf.

[113] A.E. Lawabni, H. Changjin, D.H.C. Du, and A.H. Tewfik. A novel update propagation module for the data provenance problem: A contemplating vision on realizing

data provenance from models to storage. In *Mass Storage Systems and Technologies, 2005. Proceedings. 22nd IEEE / 13th NASA Goddard Conference on*, pages 61–69, april 2005. doi:10.1109/MSST.2005.2.

[114] T. Lebo, S. Sahoo, D. McGuinness, K. Belhajjame, J. Cheney, D. Corsar, D. Garijo, S. Soiland-Reyes, S. Zednik, and J. Zhao. Prov-o: The prov ontology. *W3C Recommendation, 30th April*, 2013. URL http://hdl.handle.net/10421/7484.

[115] H. Li. Restful web service frameworks in java. In *Signal Processing, Communications and Computing (ICSPCC), 2011 IEEE International Conference on*, pages 1–4, Sept 2011. doi:10.1109/ICSPCC.2011.6061739.

[116] J. Li, X. Chen, Q. Huang, and D.S Wong. Digital provenance: Enabling secure data forensics in cloud computing. *Future Generation Computer Systems*, 37:259–266, 2014. ISSN 0167-739X. doi:http://dx.doi.org/10.1016/j.future.2013.10.006. URL http://www.sciencedirect.com/science/article/pii/S0167739X13002161.

[117] J.M. Li, G.S. Ma, G. Feng, and Y.Q. Ma. Research on web application of struts framework based on mvc pattern. In H.T. Shen, J. Li, M. Li, J. Ni, and W. Wang, editors, *Advanced Web and Network Technologies, and Applications*, volume 3842 of *Lecture Notes in Computer Science*, pages 1029–1032. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-31158-4. doi:10.1007/11610496_143. URL http://dx.doi.org/10.1007/11610496_143.

[118] X. Li, H. Zhang, and Y. Zhang. Deploying mobile computation in cloud service. In MartinGilje Jaatun, Gansen Zhao, and Chunming Rong, editors, *Cloud Computing*, volume 5931 of *Lecture Notes in Computer Science*, pages 301–311. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-10664-4. doi:10.1007/978-3-642-10665-1_27. URL http://dx.doi.org/10.1007/978-3-642-10665-1_27.

[119] M. Lorch, S. Proctor, R. Lepro, D. Kafura, and S. Shah. First experiences using xacml for access control in distributed systems. In *Proceedings of the 2003 ACM Workshop on XML Security*, XMLSEC '03, pages 25–37, New York, NY, USA, 2003. ACM. ISBN 1-58113-777-X. doi:10.1145/968559.968563. URL http://doi.acm.org/10.1145/968559.968563.

[120] J. Lu, G. Dai, D. Mu, J. Yu, and H. Li. Qos guarantee in tomcat web server: A feedback control approach. In *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2011 International Conference on*, pages 183–189, Oct 2011. doi:10.1109/CyberC.2011.39.

[121] L. Lymberopoulos, E. Lupu, and M. Sloman. An adaptive policy-based framework for network services management. *Journal of Network and Systems Management*, 11(3):277–303, 2003. ISSN 1064-7570. doi:10.1023/A:1025719407427. URL http://dx.doi.org/10.1023/A%3A1025719407427.

[122] Y. Ma, S. Jang, and J. Lee. Ontology-based resource management for cloud computing. In NgocThanh Nguyen, Chong-Gun Kim, and Adam Janiak, editors, *Intelligent Information and Database Systems*, volume 6592 of *Lecture Notes in Computer Science*, pages 343–352. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-20041-0. doi:10.1007/978-3-642-20042-7_35. URL http://dx.doi.org/10.1007/978-3-642-20042-7_35.

[123] P. Macko, M. Chiarini, M. Seltzer, and S. Harvard. Collecting provenance via the xen hypervisor. *USENIX Association*, 2011. URL http://static.usenix.org/event/tapp11/tech/final_files/MackoChiariniSeltzer.pdf.

[124] K. Mahbub and G. Spanoudakis. A framework for requirents monitoring of service based systems. In *Proceedings of the 2Nd International Conference on Service Oriented Computing*, ICSOC '04, pages 84–93, New York, NY, USA, 2004. ACM. ISBN 1-58113-871-7. doi:10.1145/1035167.1035181. URL http://doi.acm.org/10.1145/1035167.1035181.

[125] M. Mannion. Using first-order logic for product line model validation. In GaryJ. Chastek, editor, *Software Product Lines*, volume 2379 of *Lecture Notes in Computer Science*, pages 176–187. Springer Berlin Heidelberg, 2002. ISBN 978-3-540-43985-1. doi:10.1007/3-540-45652-X_11. URL http://dx.doi.org/10.1007/3-540-45652-X_11.

[126] E. Marilly, O. Martinot, S. Betge-Brezetz, and G. Delegue. Requirements for service level agreement management. In *IP Operations and Management, 2002 IEEE Workshop on*, pages 57–62, 2002. doi:10.1109/IPOM.2002.1045756.

[127] A. Martin, J. Lyle, and C. Namilkuo. Provenance as a security control. *TaPP. USENIX*, 2012. URL https://www.usenix.org/system/files/conference/tapp12/tapp12-final17.pdf.

[128] R.C. Martin. Java and c++ a critical comparison. *Technical Note, Object Mentor*, 1997. URL http://buiduychien.vnweblogs.com/gallery/4908/7.pdf.

[129] B. McBride. The resource description framework (rdf) and its vocabulary description language rdfs. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 51–65. Springer Berlin Heidelberg, 2004. ISBN 978-3-662-11957-0. doi:10.1007/978-3-540-24750-0_3. URL http://dx.doi.org/10.1007/978-3-540-24750-0_3.

[130] W. Meier. exist: An open source native xml database. In AkmalB. Chaudhri, Mario Jeckle, Erhard Rahm, and Rainer Unland, editors, *Web, Web-Services, and Database Systems*, volume 2593 of *Lecture Notes in Computer Science*, pages 169–183. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-00745-6. doi:10.1007/3-540-36560-5_13. URL http://dx.doi.org/10.1007/3-540-36560-5_13.

[131] W. Meier. Index-driven xquery processing in the exist xml database. In *a conference on XML*, page 21, 2006. URL http://archive.xmlprague.cz/2006/images/xmlprague2006.pdf#page=21.

[132] P. Mell and T. Grance. The nist definition of cloud computing (draft). *NIST special publication*, 800:145, 2011. URL http://csrc.nist.gov/publications/nistpubs/800-145.

[133] J.J.C. Meyer and R.J. Wieringa. Deontic logic: A concise overview. In J-J. C. Meyer and R. J. Wieringa, editors, *Deontic Logic in Computer Science: Normative System Specification*, pages 3–16. John Wiley & Sons, Chichester, UK, 1993. ISBN 0471937436. URL http://eprints.eemcs.utwente.nl/10664/.

[134] J. Miano. *Compressed image file formats: Jpeg, png, gif, xbm, bmp*. Addison-Wesley Professional, 1999.

[135] D. Michaelides, S. Buckingham Shum, B. Juby, C. Mancini, R. Slack, M. Bachler, R. Procter, M. Daw, A. Rowley, T. Chown, D. De Roure, and T. Hewitt. Memetic: Semantic meeting memory. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2006. WETICE '06. 15th IEEE International Workshops on*, pages 382–387, june 2006. doi:10.1109/WETICE.2006.47.

[136] P. Missier, K. Belhajjame, and J. Cheney. The w3c prov family of specifications for modelling provenance metadata. In *Proceedings of the 16th International Conference on Extending Database Technology*, EDBT '13, pages 773–776, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1597-5. doi:10.1145/2452376.2452478. URL http://doi.acm.org/10.1145/2452376.2452478.

[137] R. Mordani, J.D Davidson, and S Boag. Java api for xml processing. *Sun Microsystems*, 2001. URL https://edelivery.oracle.com/otn-pub/jcp/7552-jaxp-1.1-fr-spec-oth-JSpec/jaxp-1_1-spec.ps.

[138] L. Moreau. Usage of provenance: A tower of babel towards a concept map. *Position paper for the Life Cycle Seminar, Mountain View, July 2006*, july 2006. URL http://users.ecs.soton.ac.uk/lavm/papers/babel.pdf.

[139] L. Moreau. The foundations for provenance on the web. *Foundations and Trends in Web Science*, 2(2-3):99–241, November 2010. doi:http://dx.doi.org/10.1561/1800000010. URL http://eprints.ecs.soton.ac.uk/21691/.

[140] L. Moreau and J. Ibbotson. Standardisation of provenance systems in service oriented architectures — white paper. Technical report, University of Southampton, March 2006. URL http://eprints.soton.ac.uk/262198/.

[141] L. Moreau, J. Bradshaw, M. Breedy, L. Bunch, P. Hayes, M. Johnson, S. Kulkarni, J. Lott, N. Suri, and A. Uszok. Behavioural specification of grid services with the

kaos policy language. In *Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on*, volume 2, pages 816–823 Vol. 2, May 2005. doi:10.1109/CCGRID.2005.1558646.

[142] L. Moreau, J. Freire, J. Futrelle, R.E. McGrath, J. Myers, and P. Paulson. The open provenance model: An overview. In Juliana Freire, David Koop, and Luc Moreau, editors, *Provenance and Annotation of Data and Processes*, volume 5272 of *Lecture Notes in Computer Science*, pages 323–326. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-89964-8. doi:10.1007/978-3-540-89965-5_31. URL http://dx.doi.org/10.1007/978-3-540-89965-5_31.

[143] L. Moreau, P. Groth, S. Miles, J. Vazquez-Salceda, J. Ibbotson, S. Jiang, S. Munroe, O. Rana, A. Schreiber, V. Tan, and L. Varga. The provenance of electronic data. *Commun. ACM*, 51(4):52–58, April 2008. ISSN 0001-0782. doi:10.1145/1330311.1330323. URL http://doi.acm.org/10.1145/1330311.1330323.

[144] L. Moreau, P. Groth, S. Miles, J. Vazquez-Salceda, J. Ibbotson, S. Jiang, S. Munroe, O. Rana, A. Schreiber, V. Tan, and L. Varga. The provenance of electronic data. *Commun. ACM*, 51(4):52–58, April 2008. ISSN 0001-0782. doi:10.1145/1330311.1330323. URL http://doi.acm.org/10.1145/1330311.1330323.

[145] L. Moreau, B. Ludaescher, I. Altintas, R.S. Barga, S. Bowers, S. Callahan, JR Chin, B. Clifford, S. Cohen, S. Cohen-Boulakia, et al. Special issue: The first provenance challenge. *Concurrency and Computation: Practice and Experience*, 20(5):409–418, 2008. ISSN 1532-0634. doi:10.1002/cpe.1233. URL http://dx.doi.org/10.1002/cpe.1233.

[146] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, et al. The open provenance model core specification (v1.1). *Future Generation Computer Systems*, 27(6):743–756, 2011. ISSN 0167-739X. doi:http://dx.doi.org/10.1016/j.future.2010.07.005. URL http://www.sciencedirect.com/science/article/pii/S0167739X10001275.

[147] L. Moreau, P. Misser, J. Cheney, and S. Soiland-Reyes. PROV-N: The Provenance Notation. W3C Recommendation REC-prov-n-20130430, World Wide Web Consortium, April 2013. URL http://www.w3.org/TR/2013/REC-prov-n-20130430/.

[148] L. Moreau, P. Missier, K. Belhajjame, R. B'Far, J. Cheney, S. Coppens, S. Cresswell, Y. Gil, P. Groth, G. Klyne, T. Lebo, J. McCusker, S. Miles, J. Myers, S. Sahoo, and C. Tilmes. PROV-DM: The PROV Data Model. W3C Recommendation REC-prov-dm-20130430, World Wide Web Consortium, April 2013. URL http://www.w3.org/TR/2013/REC-prov-dm-20130430/.

[149] L. Moreau, T.D. Huynh, and D. Michaelides. An online validator for provenance: Algorithmic design, testing, and api. In Stefania Gnesi and Arend Rensink, editors, *Fundamental Approaches to Software Engineering*, volume 8411 of *Lecture Notes in Computer Science*, pages 291–305. Springer Berlin Heidelberg, 2014. ISBN 978-3-642-54803-1. doi:10.1007/978-3-642-54804-8_20. URL http://dx.doi.org/10.1007/978-3-642-54804-8_20.

[150] T. Moses et al. Extensible access control markup language (xacml) version 2.0. *Oasis Standard*, 200502, 2005. URL http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf.

[151] P. Mouallem, R. Barreto, S. Klasky, N. Podhorszki, and M. Vouk. Tracking files in the kepler provenance framework. In Marianne Winslett, editor, *Scientific and Statistical Database Management*, volume 5566 of *Lecture Notes in Computer Science*, pages 273–282. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-02278-4. doi:10.1007/978-3-642-02279-1_21. URL http://dx.doi.org/10.1007/978-3-642-02279-1_21.

[152] M.J. Moyer and M. Abamad. Generalized role-based access control. In *Distributed Computing Systems, 2001. 21st International Conference on.*, pages 391–398, apr 2001. doi:10.1109/ICDSC.2001.918969.

[153] I. Muhammad and Helmut H. Provenance in the cloud: Why and how? In *Third International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING 2012)*, USA, July 2012. URL http://eprints.cs.univie.ac.at/3554/.

[154] A. Muller, D. Happe, and S. Wilson. *Virtualization with VMware ESX server*. Syngress Media Incorporated, first edition, 2005.

[155] P. Muncaster. Gartner predicts silver lining for cloud computing. http://www.v3.co.uk/v3-uk/news/2003148/gartner-predicts-silver-lining-cloud-computing, 2009.

[156] K.K. Muniswamy-Reddy, D.A. Holland, U. Braun, and M. Seltzer. Provenance-aware storage systems. In *Proceedings of the 2006 USENIX Annual Technical Conference*, pages 43–56, 2006. URL https://www.usenix.org/legacy/event/usenix06/tech/full_papers/muniswamy-reddy/muniswamy-reddy_html/.

[157] K.K. Muniswamy-Reddy, P. Macko, and M. Seltzer. Provenance for the cloud. In *Proceedings of the 8th USENIX Conference on File and Storage Technologies*, FAST'10, pages 15–14, Berkeley, CA, USA, 2010. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1855511.1855526.

[158] S. Munroe, S. Miles, P. Groth, S. Jiang, V. Tan, L. Moreau, J. Ibbotson, and J. Vazquez-Salceda. Prime: A methodology for developing provenance-

aware applications. Technical report, University of Southampton, 2006. URL http://eprints.soton.ac.uk/263215/.

[159] S. Munroe, S. Miles, L. Moreau, and J. Vázquez-Salceda. Prime: A software engineering methodology for developing provenance-aware applications. In *Proceedings of the 6th International Workshop on Software Engineering and Middleware*, SEM '06, pages 39–46, New York, NY, USA, 2006. ACM. ISBN 1-59593-585-1. doi:10.1145/1210525.1210535. URL http://doi.acm.org/10.1145/1210525.1210535.

[160] S. Muthukumar and N. Muthu. The indian kaleidoscope: emerging trends in m-commerce. *International Journal of Advanced Research in Computer and Communication Engineering*, 4(1):50, 2015. URL http://ijarcce.com/upload/2015/january/IJARCCE1J.pdf.

[161] BBC News. Privacy and trust up in the cloud. http://news.bbc.co.uk/1/hi/programmes/click_online/8625625.stm, 2010.

[162] D. Nguyen, J. Park, and R. Sandhu. A provenance-based access control model for dynamic separation of duties. In *Privacy, Security and Trust (PST), 2013 Eleventh Annual International Conference on*, pages 247–256, July 2013. doi:10.1109/PST.2013.6596060.

[163] D. Nguyen, J. Park, and R. Sandhu. Adopting provenance-based access control in openstack cloud iaas. In ManHo Au, Barbara Carminati, and C.-C.Jay Kuo, editors, *Network and System Security*, volume 8792 of *Lecture Notes in Computer Science*, pages 15–27. Springer International Publishing, 2014. ISBN 978-3-319-11697-6. doi:10.1007/978-3-319-11698-3_2. URL http://dx.doi.org/10.1007/978-3-319-11698-3_2.

[164] Q. Ni, S. Xu, E. Bertino, R. Sandhu, and W. Han. An access control language for a general provenance model. In Willem Jonker and Milan Petkovi, editors, *Secure Data Management*, volume 5776 of *Lecture Notes in Computer Science*, pages 68–88. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-04218-8. doi:10.1007/978-3-642-04219-5_5. URL http://dx.doi.org/10.1007/978-3-642-04219-5_5.

[165] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The eucalyptus open-source cloud-computing system. In *Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on*, pages 124–131, May 2009. doi:10.1109/CCGRID.2009.93.

[166] Orange. Orange group. http:www.orange.com/en_EN/group, 2010.

[167] Orange. Orange business services continues momentum in cloud computing market. http://www.orange.com/en/press/Press-releases/press-releases-2013/

Orange-Business-Services-continues-momentum-in-cloud-computing-market, 2013.

[168] Labs Orange. Confidenshare service. http://labs.orange.com/en/work/projects/ConfidenShare, 2015.

[169] M.R. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel. Amazon s3 for science grids: A viable solution? In *Proceedings of the 2008 International Workshop on Data-aware Distributed Computing*, DADC '08, pages 55–64, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-154-5. doi:10.1145/1383519.1383526. URL http://doi.acm.org/10.1145/1383519.1383526.

[170] J. Park, D. Nguyen, and R. Sandhu. A provenance-based access control model. In *Privacy, Security and Trust (PST), 2012 Tenth Annual International Conference on*, pages 137–144, July 2012. doi:10.1109/PST.2012.6297930.

[171] C. Pautasso. {RESTful} web service composition with {BPEL} for {REST}. *Data & Knowledge Engineering*, 68(9):851–866, 2009. ISSN 0169-023X. doi:http://dx.doi.org/10.1016/j.datak.2009.02.016. URL http://www.sciencedirect.com/science/article/pii/S0169023X09000366. Sixth International Conference on Business Process Management (BPM 2008) Five selected and extended papers.

[172] A. Paventhan, K. Takeda, S.J. Cox, and D.A. Nicole. Leveraging windows workflow foundation for scientific workflows in wind tunnel applications. In *Data Engineering Workshops, 2006. Proceedings. 22nd International Conference on*, pages 65–65, 2006. doi:10.1109/ICDEW.2006.71.

[173] S. Pearson. Toward accountability in the cloud. *Internet Computing, IEEE*, 15 (4):64–69, July 2011. ISSN 1089-7801. doi:10.1109/MIC.2011.98.

[174] J. Peng, X. Zhang, Z. Lei, B. Zhang, W. Zhang, and Q. Li. Comparison of several cloud computing platforms. In *Information Science and Engineering (ISISE), 2009 Second International Symposium on*, pages 23–27, Dec 2009. doi:10.1109/ISISE.2009.94.

[175] E. Pignotti, P. Edwards, A. Eckhardt, and K. Ponnamperuma. A toolkit for developing provenance-aware applications. 2012. URL http://www.dotrural.ac.uk/digitalfutures/sites/default/files/digitalfutures2012papers/Demos/Pignotti_etal_Toolkit.pdf.

[176] P. Pinheiro da Silva and S. Roach. A Use Case-Guided Comparison of OPM and PML. *Submitted to Journal of Web Semantics*, 2010. URL http://digitalcommons.utep.edu/cs_techrep/669/.

[177] M.T.E.T.E. Potok and F.T. Sheldon. Dynamic data fusion using an ontology-based software agent system. *Proceedings of the IIIS Agent Based Computing, Orlando*, 7, 2003. URL http://cda.ornl.gov/publications/SCI03-Elmore.pdf.

[178] E. PrudHommeaux, A. Seaborne, et al. SPARQL Query Language for RDF. Technical report. URL http://www.w3.org/TR/rdf-sparql-query/.

[179] H. Pundt and Y. Bishr. Domain ontologies for data sharingan example from environmental monitoring using field {GIS}. *Computers & Geosciences*, 28(1):95–102, 2002. ISSN 0098-3004. doi:http://dx.doi.org/10.1016/S0098-3004(01)00018-8. URL http://www.sciencedirect.com/science/article/pii/S0098300401000188.

[180] W. Quan-bin. Application of mvc framework in the development of java [j]. *Journal of Sichuan University of Science & Engineering (Natural Science Edition)*, 1:012, 2009.

[181] M. Ramane, B. Vasudevan, and S. Allaphan. A provenance-policy based access control model for data usage validation in cloud. *CoRR*, abs/1411.1933, 2014. URL http://arxiv.org/abs/1411.1933.

[182] P. Reddivari, T. Finin, and A. Joshi. Policy-based access control for an rdf store. *Policy Management for the Web*, pages 78–81, 2005. URL http://ebiquity.umbc.edu/get/a/publication/323.pdf.

[183] E. Rescorla. *SSL and TLS: designing and building secure systems*, volume 1. Addison-Wesley Reading, 2001.

[184] C. Ringelstein and S. Staab. Papel: A language and model for provenance-aware policy definition and execution. In Richard Hull, Jan Mendling, and Stefan Tai, editors, *Business Process Management*, volume 6336 of *Lecture Notes in Computer Science*, pages 195–210. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15617-5. doi:10.1007/978-3-642-15618-2_15. URL http://dx.doi.org/10.1007/978-3-642-15618-2_15.

[185] E. Rissanen. extensible access control markup language (xacml) version 3.0 (committe specification 01). Technical report, Technical report, OASIS, 2010. URL http://docs.oasisopen.org/xacml/3.0/xacml-3.0-core-spec-cd-03-en.pdf.

[186] E Rissanen et al. extensible access control markup language (xacml) version 3.0, 2013.

[187] J.W. Rittinghouse and J.F. Ransome. *Cloud computing: implementation, management, and security*. CRC, 2009.

[188] K. Ruan, J. Carthy, T. Kechadi, and M. Crosbie. Cloud forensics definitions and critical criteria for cloud forensic capability: An overview of survey results. *Digital Investigation*, 10(1):34–43, 2013. ISSN 1742-2876. doi:http://dx.doi.org/10.1016/j.diin.2013.02.004. URL http://www.sciencedirect.com/science/article/pii/S1742287613000121.

[189] K Russell. Quick study: Extensible access control markup language (xacml). *Computerworld*, 104841:19, 2003. URL http://www.computerworld.com/s/article/81295/XACML.

[190] S. Sahoo, P. Groth, O. Hartig, S. Miles, S. Coppens, J. Myers, Y. Gil, L. Moreau, J. Zhao, M. Panzer, and D. Garijo. Provenance vocabulary mappings. http://www.w3.org/2005/Incubator/prov/wiki/Provenance_Vocabulary_Mappings, 2010.

[191] S.S. Sahoo and A. Sheth. Provenir ontology: Towards a framework for escience provenance management. In *Microsoft eScience Workshop, Pittsburgh, PA, USA*, 2009. URL http://corescholar.libraries.wright.edu/knoesis/80/.

[192] M.A. Sakka, B. Defude, and J. Tellez. Document provenance in the cloud: Constraints and challenges. In FinnArve Aagesen and SveinJohan Knapskog, editors, *Networked Services and Applications - Engineering, Control and Management*, volume 6164 of *Lecture Notes in Computer Science*, pages 107–117. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-13970-3. doi:10.1007/978-3-642-13971-0_11. URL http://dx.doi.org/10.1007/978-3-642-13971-0_11.

[193] B.K. Samanthula, Y. Elmehdwi, G. Howser, and S. Madria. A secure data sharing and query processing framework via federation of cloud computing. *Information Systems*, 48:196–212, 2015. ISSN 0306-4379. doi:http://dx.doi.org/10.1016/j.is.2013.08.004. URL http://www.sciencedirect.com/science/article/pii/S0306437913001208.

[194] R.S. Sandhu and P. Samarati. Access control: principle and practice. *Communications Magazine, IEEE*, 32(9):40–48, sept. 1994. ISSN 0163-6804. doi:10.1109/35.312842.

[195] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996. ISSN 0018-9162. doi:http://doi.ieeecomputersociety.org/10.1109/2.485845.

[196] K. Schwaber. *Agile project management with Scrum*. O'Reilly Media, Inc., 2009.

[197] P. Sempolinski and D. Thain. A comparison and critique of eucalyptus, opennebula and nimbus. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 417–426, Nov 2010. doi:10.1109/CloudCom.2010.42.

[198] Y. Simmhan, B. Plale, D. Gannon, and S. Marru. Performance evaluation of the karma provenance framework for scientific workflows. In Luc Moreau and Ian Foster, editors, *Provenance and Annotation of Data*, volume 4145 of *Lecture Notes in Computer Science*, pages 222–236. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-46302-3. doi:10.1007/11890850_23. URL http://dx.doi.org/10.1007/11890850_23.

[199] A. Smith. Smartphone ownership–2013 update. *Pew Research Center: Washington DC*, 12, 2013. URL http://boletines.prisadigital.com/PIP_Smartphone_adoption_2013.pdf.

[200] D.M. Smith. Cloud computing security & management. http://www.gartner.com/technology/research/cloud-computing/cloud-security-management.jsp, 2010.

[201] E. Smith. Cloud computing: The essential foundation of industry digital transformation worldwide and u.s. cloud forecast by vertical, 20152019. https://www.idc.com/getdoc.jsp?containerId=256944, 2015.

[202] S. Spielman. *The Struts Framework: Practical Guide for Java Programmers*. Morgan Kaufmann, 2002.

[203] D.D. Steinauer, S.A. Wakid, and S. Rasberry. Trust and traceability in electronic commerce. *StandardView*, 5(3):118–124, September 1997. ISSN 1067-9936. doi:10.1145/266231.266239. URL http://doi.acm.org/10.1145/266231.266239.

[204] B. Stepien, S. Matwin, and A. Felty. Advantages of a non-technical xacml notation in role-based models. In *Privacy, Security and Trust (PST), 2011 Ninth Annual International Conference on*, pages 193–200, July 2011. doi:10.1109/PST.2011.5971983.

[205] L. Stotler. Ovum survey indicates major cloud services uptake; telecom providers rated as trusted partners. http://www.mspnews.com/msp/articles/175833-ovum-survey-indicates-major-cloud-services-uptake-telecom.htm, 2011.

[206] L. Sun, J. Park, and R. Sandhu. Engineering access control policies for provenance-aware systems. In *Proceedings of the Third ACM Conference on Data and Application Security and Privacy*, CODASPY '13, pages 285–292, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1890-7. doi:10.1145/2435349.2435390. URL http://doi.acm.org/10.1145/2435349.2435390.

[207] A. Syalim, Y. Hori, and K. Sakurai. Grouping provenance information to improve efficiency of access control. In JongHyuk Park, Hsiao-Hwa Chen, Mohammed

Atiquzzaman, Changhoon Lee, Tai-hoon Kim, and Sang-Soo Yeo, editors, *Advances in Information Security and Assurance*, volume 5576 of *Lecture Notes in Computer Science*, pages 51–59. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-02616-4. doi:10.1007/978-3-642-02617-1_6. URL http://dx.doi.org/10.1007/978-3-642-02617-1_6.

[208] T. Thai and H. Lam. . *.NET framework essentials*. "O'Reilly Media, Inc.", 2003.

[209] W.T. Tsai, W. Xiao, Z. Dawei, R. Paul, C. Yinong, and J.-Y. Chung. A new soa data-provenance framework. In *Autonomous Decentralized Systems, 2007. ISADS '07. Eighth International Symposium on*, pages 105–112, March 2007. doi:10.1109/ISADS.2007.5.

[210] S. Tunnicliffe and I. Davis. Changeset vocabulary, 2005. URL http://purl.org/vocab/changeset.

[211] R. Uhlig, G. Neiger, D. Rodgers, A.L. Santoni, F.C.M. Martins, A.V. Anderson, S.M. Bennett, A. Kagi, F.H. Leung, and L. Smith. Intel virtualization technology. *Computer*, 38(5):48–56, May 2005. ISSN 0018-9162. doi:10.1109/MC.2005.163.

[212] J. Urquhart. The biggest cloud-computing issue of 2009 is trust. http://news.cnet.com/8301-19413_3-10133487-240.html, 2009.

[213] A. Uszok, J.M. Bradshaw, M. Johnson, R. Jeffers, A. Tate, J. Dalton, and S. Aitken. Kaos policy management for semantic web services. *Intelligent Systems, IEEE*, 19(4):32–41, aug 2004. ISSN 1541-1672. doi:10.1109/MIS.2004.31.

[214] C. Vecchiola, X. Chu, and R. Buyya. Aneka: a software platform for .net-based cloud computing. *High Speed and Large Scale Scientific Computing*, 18:267–295, 2009. URL http://arxiv.org/pdf/0907.4622.pdf?pagewanted=all.

[215] J. Voas and J. Zhang. Cloud computing: New wine or just a new bottle? *IT Professional*, 11(2):15–17, 2009. ISSN 1520-9202. doi:http://doi.ieeecomputersociety.org/10.1109/MITP.2009.23.

[216] E. Walker. Benchmarking amazon ec2 for high-performance scientific computing. *Usenix Login*, 33(5):18–23, 2008.

[217] D.J. Weitzner, H. Abelson, T. Berners-Lee, J. Feigenbaum, J. Hendler, and G.J. Sussman. Information accountability. *Commun. ACM*, 51(6):82–87, June 2008. ISSN 0001-0782. doi:10.1145/1349026.1349043. URL http://doi.acm.org/10.1145/1349026.1349043.

[218] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):560–576, July 2003. ISSN 1051-8215. doi:10.1109/TCSVT.2003.815165.

[219] F. Xu, A. Biller, L. Chen, V. Tan, P. Groth, S. Miles, J. Ibbotson, and L. Moreau. A proof of concept design for provenance. Technical report, University of Southampton, March 2005. URL http://eprints.soton.ac.uk/260687/.

[220] L. Youseff, M. Butrico, and D. Da Silva. Toward a unified ontology of cloud computing. In *Grid Computing Environments Workshop, 2008. GCE '08*, pages 1–10, Nov 2008. doi:10.1109/GCE.2008.4738443.

[221] E. Yuan and J. Tong. Attributed based access control (abac) for web services. In *Web Services, 2005. ICWS 2005. Proceedings 2005 IEEE International Conference on*, pages 2 vol. (xxxiii+856), july 2005. doi:10.1109/ICWS.2005.25.

[222] O.Q. Zhang, R.K.L. Ko, M. Kirchberg, Chun Hui Suen, P. Jagadpramana, and Bu Sung Lee. How to track your data: Rule-based data provenance tracing algorithms. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*, pages 1429–1437, June 2012. doi:10.1109/TrustCom.2012.175.

[223] L. Zhi, W. Jing, C. Xiao-su, and J. Lian-xing. Research on policy-based access control model. In *Networks Security, Wireless Communications and Trusted Computing, 2009. NSWCTC '09. International Conference on*, volume 2, pages 164–167, april 2009. doi:10.1109/NSWCTC.2009.313.