Preserving Records in the Cloud – A Model to enhance Metadata Interoperability in a Cloud Environment

September 2013

Jan Askhoj

Preserving Records in the Cloud – A Model to enhance Metadata Interoperability in a Cloud Environment

Jan Askhoj

Graduate School of Library, Information and Media Studies

University of Tsukuba

September 2013

Preserving Records in the Cloud – A Model to enhance Metadata Interoperability in a Cloud Environment

Abstract

An increasing number of organisations are using cloud computing to create and store digital records. The problems relating to the preservation of electronic documents in general are well known, and steps can be taken to ensure systems can provide long-term accessibility and readability of electronic records. However, with cloud computing, the management and responsibility of infrastructure, systems and data may no longer reside in the organisation in which the electronic records are created. For this reason, many producers of digital contents choose to transfer these to a dedicated archive. However, since such a transfer can be both costly and time-consuming, this raises the question of how the process can be simplified and what can be done to increase interoperability between producer and archive, when one or both of these are in the cloud.

This thesis examines cloud computing from an archiving perspective and how this new technology fits with existing models of digital archiving, exemplified by the Open Archival Information System (OAIS) reference model. In OAIS, digital archives receive their records and accompanying metadata from a producing organisation or institution in a predetermined package format. The archive and producer agree on a set of requirements for submission, defined by the archive in order to ensure an easy ingest. Once the contents have been ingested, they are stored and managed in a data centre, where the archive has complete control of the technological infrastructure and digital objects. Using this infrastructure and digital archiving software, the archive provides access to archive users, while ensuring the ongoing preservation of the archiving collection.

Based on the above reference model, the thesis identifies four areas where the OAIS model does not address the requirements of a cloud environment. 1) The fact that the functional entities in OAIS are interdependent, makes it difficult to transfer responsibility for parts of an OAIS archive to an external service provider. For example, if an organisation is looking for a storage solution offering bit-level integrity for digital objects to use as a back-end for an archiving

system, this would involve overlapping functionality from the Management, Data Archiving and Archival Storage entities. 2) In OAIS, the burden of creating Submission Information Packages (SIP) is left to producers, who must meet the requirements of the OAIS archive. Many of these requirements are related to metadata. An archive will specify a number of mandatory metadata elements that must be included in SIPs and must comply with the formatting and schema rules for submission packages. For a producer, complying with this can be very resource intensive, depending on the strictness of the requirements. This can lead to producers holding on to records for long periods, before submitting them in bulk, which can significantly delay preservation planning. 3) With cloud computing, there is less need to include digital objects Information Packages With a shared and trusted platform, producers only need to provide the information (URI or similar) of where the digital objects are stored. However, the OAIS Model does not specify the requirements and functionality of such a shared platform. 4) The OAIS model does not cover the initial stages of the Document Lifecycle (i.e., the Create, Use, Manage stages). It can be argued that these stages lie outside the scope of an archive. However, the nature of the events in these stages and how well they are documented can have a huge impact on how easy it will be to carry out preservation work later on.

Based on the findings of the examination, a model for a cloud archiving system to improve interoperability between Producer and Archive is proposed using concepts and information types from OAIS. The information that comprises an OAIS information Package can be arranged according to complexity. There is an increase in complexity from the simple digital object to the comprehensive Information Package. This progression from simple to complex is comparable to how information flows in a layered model, where information in one layer is used, manipulated and passed to a higher layer. This model reflects the development in complexity of digital objects, and is similar to the document lifecycle, where a document goes through a number of stages over time. The proposed model allows the sharing of functionality and digital objects by making these available as services to above layers. The model covers the entire document lifecycle, making archive functionality such as preservation planning possible at an early stage in the document lifecycle and helps to simplify records transfer. The model is explained in a theoretical case study, using the records transfer process from Japanese government agencies to the National

Archives of Japan as an example.

Whereas the proposed layered model serves as a basic conceptual model, it does not solve the problem of how SIPs should be structured. To describe the metadata that should be included in SIPs and where in the layered model it originates, the thesis proposes a metadata application profile for cloud archives. As interoperability is an essential part of the proposed cloud solution (referring here not only to interoperability between producer and archive but also to potential interoperability between different digital archives), the author chose to design the application profile using the Singapore Framework for Dublin Core Application Profiles to define the functional requirements, domain model and description set profile that form the basis of the proposed application profile. In the profile, METS is used as a transmission and package format, extending it with metadata from the PREMIS data dictionary and Dublin Core Metadata Element Set. METS was chosen for a number of reasons: It allows the inclusion of other metadata schemas, it can express structurally complex objects and several solutions using METS already exist. PREMIS defines core preservation metadata (semantic units) needed to support long-term preservation. Using the proposed application profile, an example METS information package is created with predefined criteria. It was found that the application profile can simplify metadata provision for business systems, compared to systems that do not allow pre-registration. Furthermore, there is a potential for the automation of metadata provision, further reducing the amount of metadata that must be explicitly provided. A further examination was performed on the metadata that must be provided by Producers and that cannot be automated. It was found that many of the elements described complicated attributes of digital objects, such as structural relations, encryption or rights information. The more complex the digital objects to be preserved, the more metadata must be provided by business systems, increasing cost for producers.

The proposed model and Application Profile answered the research questions dealing with how a model can be developed in such a way that it integrates the requirements of both producer and archive when building a cloud-based digital archive and what such a system would look like. However, one major barrier to actual implementation is the lack of a formal semantic model and common vocabulary expressed in a machine-readable format. The thesis proposes an

OWL ontology for cloud archive systems built on the Library of Congress PREMIS ontology, combined with the layered model of cloud computing. It defines classes and their allowable domains, ranges and properties and provides a semantic framework that allows linking to metadata from other schemas.

The author believes that the strength of the ontology lies in the fact that it not only describes a metadata model for Information Packages, but also for the entities contributing to these packages. This is important in an environment like the cloud, where the sharing of computing resources (such as storage) is common, and where different information generating entities may not be capable of supplying Submission Packages in a format defined by an archive.

The ontology was evaluated with a prototype system, using real-world examples of cloud systems and digital objects. It was evident that the system contained a high degree of complexity. There are a number of factors contributing to this: multiple steps, multiple data sources, an extensive metadata set and strict requirements for metadata quality. This can lead to a greater margin for error in system and archival metadata. The OWL ontology was used to perform validation on SIP metadata saved as RDF/XML using SPARQL. The ontology turned out to be a powerful tool, not only in identifying incorrect metadata, but also to point out the origin of that metadata in order to correct the problem and preventing it from reoccurring.

It was found that the ontology was able to describe the chosen components successfully, and that it improved metadata interoperability between content creating applications and the services providing preservation metadata.

The thesis creates a theoretical framework for the building of digital cloud archives based on a layered model of services. Using this approach, it becomes possible to abstract and negotiate levels of service between producer and archive, while still guaranteeing bit-level data integrity. The concepts from the layered model are used in the creation of an application profile and ontology. Using case studies with real-world systems and data, the author demonstrates that the model can be implemented in practice. When doing so, the model is shown to improve interoperability between producer and archive by enabling the sharing of resources, automated submission of digital objects and metadata validation.

クラウドにおける記録保存 – クラウド環境におけるメタデータ相互運用性向上のためのモデル

概要

クラウドコンピューティングを活用して電子記録を作成し保管する組織が急速に増えている。電子文書の保存に関する問題があることが一般によく知られている。この問題に対して、電子的な記録文書へのアクセスと可読性を長期に渡って保つことができるようにシステムを実現することは可能である。しかしクラウドコンピューティングの場合では、インフラやシステム、データの管理等の重要な事項に関する責任が、電子記録を作成した組織ではなく、クラウドを提供する組織に移ることになる。そのため、外部にゆだねることを不安に思う記録文書の作成者の多くが専用アーカイブの開発を選択することになってしまう。こうした不安を解消するには、記録文書をクラウド上に置いて管理するために、作成者とクラウドの間での相互運用性を持つ、クラウド上における文書保存のための文書の移管と保存に関するメタデータを明確に定義することが必要である。

本論文では、クラウドコンピューティングをアーカイビングの視点から検証しOpen Archival Information System (OAIS)参照モデルに代表される既存のデジタルアーカイビングの中に、本研究で提案した新しい技術がどのように適用できるかについて検討した。OAIS では、デジタルアーカイブが、記録とそれに付随するメタデータを作成した組織から事前に決められたパッケージ形式で受け取るとしている。アーカイブが決めるコンテンツの提出要件に関しアーカイブと作成者が合意することで、アーカイブにおけるコンテンツ受領を容易に進めることができる。一旦取り込まれたコンテンツは、アーカイブが管理するデータセンターでは、技術的インフラの上にデジタルオブジェクトとして保存・管理される。このデータセンターでは、利用者がアーカイブに蓄積されたコンテンツにアクセスできるようにすると同時に、アーカイブされたコンテンツのコレクションを長期間保存することができる。

こうした従来の方法に対して、本論文ではOAIS モデルがクラウド環境の要件を

満たさない以下の４つのケースを明らかにした。（１）OAIS における機能的な実体が相互に依存しているため、そうした機能実体の一部を外部のサービス提供者に任せることが困難である。例えば、ある組織がアーカイブシステムのバックエンドとして使う目的でビットレベルの整合性を有するデジタル保存方法を求めている場合、OAIS に定義されているマネジメントとデータアーカイブ、アーカイブストレージの 3 つの機能実体が、外部に作られるバックエンド機能と重複することになる。（２）OAIS では、OAIS アーカイブの要件を満たす提出用情報パッケージ（SIP)を作るという負担が作成者にかかる。これらの要件の大部分はメタデータに関連している。提出パッケージのフォーマットと提出要件に関わるメタデータ要素はアーカイブが指定する。提出要件の厳格さなどにもよるが、提出者がアーカイブから求められる要件を満たすには膨大な量の作業が必要とされる。そのため、作成者が大量の記録を提出する前に長期に渡ってそれらを手元に置いてしまうことにつながり、保存計画を顕著に遅らせることにもなりかねない。（３）共有かつ信頼できるプラットフォームとしてのクラウドコンピューティング環境では、デジタルオブジェクトとメタデータを一つの情報パッケージに含める必要性がなくなり、作成者は URI か類似の情報を提供するだけでよい。しかし、OAIS モデルはそういった共有プラットフォームを前提として作られたものではないため、プラットフォームに求められる要件と機能を特定していない。（４）OAIS モデルは、作成・利用・管理というドキュメントサイクルの初期段階を網羅していない。これらの段階はアーカイブの機能外であると言えるかもしれないが、これらの初期段階におけるコンテンツに関する情報を適切に文書化することが、その後の保存作業の難易度に大きく影響する。

デジタル保存においては、コンテンツとその保存に関わる様々な情報をメタデータとして記録するため、メタデータをコンテンツ作成者とアーカイブの間で適切に流通させる必要がある。本論文では、前述の考察結果を踏まえ、OAIS が定義する種々の概念と情報タイプを基盤として、クラウド環境を利用するコンテンツの作成者とクラウド上に置かれるアーカイブの間でのメタデータの相互運用性を向上するクラウドアーカイビングシステムのモデルを提案する。OAIS の基本情報タイプは、比較的単純なデジタルオブジェクトから、より包括的な情報パッケージへと複雑化する。OAIS モデルが持つこの特徴を、クラウド

が持つ機能階層に置き換えると、一つの層における情報が抽象化されて上位の層に渡され、利用されるという階層モデルにおける情報の流れに対応づけて理解することができる。文書のライフサイクルに対応して保存管理のためのメタデータが追加されていくということに基づき、このモデルは、デジタルオブジェクトがいくつもの段階を通して保存のために構造化されていく過程を反映している。このモデルは文書のライフサイクル全体を網羅し、ライフサイクル初期における保存管理計画等の策定を可能にし、さらにクラウド環境を利用した記録文書の移行の簡素化に役立つ。なお、こうした特色を検証するため、本論文では、日本の政府組織から国立公文書館への文書移管プロセスを適用事例として提案モデルの検討を行った。

ここで提案しているレイヤードモデルは基本的な概念モデルであるが、SIP をどのように構築するかという問題の答えにはならない。SIP に含まれるべきメタデータと、それがレイヤードモデルのどこから発生するかを説明するため、本論文は、クラウドアーカイブ向けのメタデータスキーマ定義のモデルを、メタデータアプリケーションプロファイル（Metadata Application Profile）の概念に基づき提案する。ここでは Dublin Core Metadata Initiative (DCMI)が提案する Singapore Framework に基づき、機能要件と Domain Model 及び Description Set Profile を定義し、本論文が提案するクラウド上でのアーカイブ実現のキーとなるメタデータの相互運用性の視点から検討している。定義した SIP のためのメタデータスキーマを登録し、アーカイブと作成者間での SIP の送受に利用する。本論文では、メタデータの送受（Transmission）とパッケージ化のためのメタデータ標準として METS を用い、その上で PREMIS data dictionary と Dublin Core Metadata Element Set を利用してメタデータ記述能力を高めた。METS を選択したのは、複数のメタデータスキーマを包含した定義が可能である、PREMIS や Dublin Core を METS 上で利用することにより、構造的に複雑なオブジェクトに対して複数の解決方法を表すことが可能である等の理由による。なお、PREMIS は、保存の核になるメタデータ記述項目を定義し、Dublin Core は汎用のコンテンツ記述のためのメタデータ記述項目を定義している。本研究では、提案したアプリケーションプロファイルを用い、事前に定義された基準に沿って情報パッケージの例を作成した。その結果、メタデータスキーマを事前登録できないシステムと比較すると、アプリケーションプロファイル

の登録によりメタデータ提供を簡素化できることが分かった。加えて、人的作業によって作成されるメタデータの量を減らし、メタデータを自動的に提供できる可能性もある。作成者側のシステムによって提供されるべきではあるが、作成自動化の難しいメタデータについてもさらに検証した。その結果、多くのメタデータ記述項目が構造的な関係や暗号化、または権利情報などのデジタルオブジェクトの複雑な属性を表現していることが明らかになった。 保存すべきデジタルオブジェクトが複雑になればなるほど、作成者側のシステムが提供するメタデータも増えることになり、作成コストも増加する。

クラウドを基礎としたデジタルアーカイブを構築する際に、作成者とデジタルアーカイブの要件の統合モデルをどのように開発するか、そしてそれがどのようなアーカイブシステムであるかという観点から研究を進め、本論文で提案しているアプリケーションプロファイルとモデルを得た。しかしながら、これを実際に運用するには、形式的セマンティックモデルと機械読取り可能なフォーマットを使った共通言語という問題がある。そのため、本論文が提案するクラウドコンピューティングの階層モデルとアプリケーションプロファイルを統合するオントロジーを、PREMIS オントロジーのためのオントロジーを基礎として定義した。

著者は、オントロジー定義の利点は情報パッケージ向けのメタデータモデルだけでなく、これらのパッケージに関連付けられた種々の実体を記述する点にあると考える。ストレージ等の資源の共有が高度に行われ、またコンテンツの実体に関わる情報が多様であるために提出パッケージ製作手続きが必ずしも統一化できないというクラウド環境においては、クラウドを構成する階層上での実体の意味的関係定義の基盤としてオントロジーは重要な役割を持つ。

クラウドシステムとデジタルオブジェクト、メタデータの実際の例を用いたプロトタイプシステムによってオントロジーを評価した結果、このシステムが高度に複雑な面を持つことが明らかになった。その要因としては、複数のステップとデータソースを持つこと、メタデータ語彙が多岐にわたること、メタデータの質に関する厳しい要件が与えられることなどが挙げられる。これら全ての要件はシステムとアーカイブメタデータにエラーが起きる可能性を陽に表している。また、OWL を用いてオントロジーを定義したことで、情報パッケージの

ためのメタデータのためにSPARQLが利用でき、多様なコンテンツのための多様なメタデータを扱うことが行いやすくなるのみならず、メタデータの誤りの指摘などのための強力なツールとなっている。このように、コンテンツ作成アプリケーションとメタデータを保管するサービスとの間のメタデータの相互運用性を高めるためにオントロジー定義が役立つことが確認できた。

本論文では、サービスのレイヤードモデルに基づいてデジタルクラウドアーカイブを構築するための理論的枠組みを作成した。このアプローチにより、ビットレベルのデータ整合性を確保しつつ、作成者とアーカイブ間のサービスレベルを抽象化し調整することが可能になる。アプリケーションプロファイルとオントロジーを作成するためレイヤードモデルの概念を使用した。実際のシステムとデータを用いた事例によって、このモデルが実際に運用しうることが証明できた。以上から、このモデルはリソースを共有し、デジタルオブジェクト提出とメタデータ検証を自動化することにより、作成者とアーカイブ間の相互運用性を改善することができると言える。

Table of Contents

Table of Figures

1 Introduction

In the last few years, cloud computing has seen rapid growth and has even entered the vocabulary of ordinary consumers, thanks in part to services offered by major technology vendors, such as Apple with its iCloud, Dropbox and Google Drive.

As cloud computing is becoming more well known, usage is increasing dramatically. A shift has started in the computing landscape, where cloud computing has become a popular choice for many organisations that wish to move data, software and sometimes the entire technical infrastructure from corporate data-centers to Cloud Service Providers. IDC predicts that by 2020 as much as 15% of the information in the Digital Universe could be part of a cloud service. Whereas much of this information may not be considered preservation worthy, a large and increasing amount of business and administrative records are being stored in the cloud (Gantz & Reinsel 2010).

There are a number of reasons why organisations are choose to utilise cloud computing. Examples include:

- **Reduced initial cost.**

Compared to traditional IT system implementation, there is generally very little initial cost associated with cloud computing. There is no need to purchase and configure new hardware, as this is all managed by the vendor. Depending on the type of solution, installation and configuration of operating system and software can be similarly avoided.

- **Reduced ongoing cost.**

Cloud computing providers are able to offer economies of scale that are hard to obtain for individual organisations. By investing in virtualisation and load basing technologies, cloud providers can ensure that their servers are running at close to 100% capacity at any given time.

- **On-demand scalability.**

Cloud computing enables the scaling of resources such as processing or storage on demand. Furthermore, many cloud providers offer usage monitoring

tools, so that organisations only pay for the resources they use.

- **No need for software deployment.**

Because cloud services are offered over the Internet, users and system administrators do not need any special software installed on client PCs. In most cases, all that is needed is a modern web browser.

- **Other benefits.**

Other benefits listed include greater flexibility, allowing IT staff to shift focus from basic to value added services, and being able to access services on a large number of different devices, including mobile device. (Miri & Mintz Testa 2011)

With such growth in content created and stored in the cloud, it becomes clear that there is a need to ensure that this information is properly stored and that its long term preservation is guaranteed. The corruption of digital contents has always been a factor of risk, but with content stored by third-party providers, the risk is further increased. Changes in provided service, improper preservation metadata or in the worst case, the complete discontinuation of services can lead to a huge loss of data. For this reason, there is a need to see how archiving and preservation of digital contexts in the cloud can be done and to introduce methods for doing this, if existing models prove insufficient. This thesis seeks to answer these question by examining the phenomenon of cloud computing from an archiving perspective.

The thesis seeks to contribute to the field of knowledge by using a layered model of cloud computing service models, as defined by the National Institute of Standards and Technology and applying it to an archiving system based either wholly or partly in the cloud. It further builds on this model by defining an application profile and ontology that defines the exact metadata classes that needs to be present at each layer. The model and application profile/ontology are validated in a series of case scenarios, showing that it is indeed possible to create a cloud based archiving system that not only provides preservation metadata, but also increases interoperability between producers of digital content and archives by greatly simplifying and automating the submission process.

2 Definition of Cloud Computing

Whereas the concept of cloud computing has started appearing in the mainstream over the last decade or so, the idea is much older. One of the first to be credited with the idea was the American Computer & Cognitive Scientist John McCarthy who in the 1960s wrote that “computation may someday be organized as a public utility.” In 1969, one of the fathers of ARPANET, J.C.R. Licklider, introduced his vision for an "intergalactic computer network". However, it wasn't until the early 1990's that this vision became a reality with the advent of grid computing, as a way of delivering distributed computing power as a utility (Mohamed 2009).

The term “Cloud Computing” appears to have been made its first formal appearance in a talk titled “Intermediaries in Cloud-Computing” presented at the INFORMS meeting in Dallas in 1997. It has its origins in the networking diagrams, where the Internet is represented as a giant cloud, signifying the presence of an unspecified number of servers and networking connections (Chellappa 1997).

The first company credited with creating what we today understand as a cloud computing service was the Customer relationship management software provider Salesforce.com. In 1999, they launched a service delivering enterprise applications via a simple website. The service proved to be a success, and Salesforce.com was soon followed by a wave of companies delivering similar services (Salesforce 2012).

Cloud computing builds on existing technologies such as the internet, client server-architecture and the use of browsers, and is similar to other ways of delivering computing resources over the a network, such as in GRID computing. In 2008, this this famously led Oracle CEO to exclaim:

“ The interesting thing about cloud computing is that we’ve redefined cloud computing to include everything that we already do. I can’t think of anything that isn’t cloud computing with all of these announcements. (Krangel 2009)

Indeed, in recent years, cloud has become a household word, with many of the worlds biggest IT brands delivering some kind of cloud services, such as Apples

iCloud, Amazons Elastic Compute Cloud, Microsoft Azure, Google Apps etc.

2.1 Towards a Formal Definition of Cloud Computing

When the author first started this research in 2009, there were a large number of definitions of cloud computing in existence. In their paper A Break in the Clouds: Towards a Cloud Definition, Vaquero et. al, compare more than 20 definitions from the published papers on computing. Some of these definitions are very general, while some are focused on specific aspects of technology, such as service delivery or virtualisation.

In order to define the application domain for the usage of cloud computing when it comes to records storage, the author initially developed his own 4 point definition of cloud computing, partly based on existing definitions (Vaquero et al. 2008):

1. **Cloud computing is an abstracted, scalable platform for service delivery.**

   Abstracted in this context means that the computing resources (i.e. the hardware hosting the cloud) are presented to the users as a unified, single resource. The hardware and individual virtual machines can be hosted on different pieces of hardware in several different geographical locations. That cloud computing is scalable means that services such as processing power or storage can be increased on demand. If an application or process is in need of extra computing resources, these can be provided on an ad-hoc basis.

2. **Cloud computing makes use of existing technologies that can be described via a layered model.**

   People have pointed out that cloud computing in itself is nothing new. It builds on pre-existing technologies such as virtualisation, grid-computing and the Internet. However, the way these technologies are connected and used for service delivery can be described as new. With services such as the Google App Engine and Amazon EC2, it is now possible for businesses and individuals to gain quick, affordable access to computing resources previously limited to large organisations with dedicated data centres.

Of the prevalent ways of distinguishing between the different levels of abstraction is by dividing them into 3 service levels: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). Using service levels as the unit of abstraction is only one of many ways of layering cloud computing service offerings.

3. **Access to both platform and services is available via the internet.** In cloud computing, access to software, data, APIs and platform administration tools is available via the internet, using a browser or similar tool. Management of the hardware that the cloud platform operates on still has to be done directly in the data centre, but for end users, no such direct access is necessary.
4. **The availability, quality and number of services are offered according to agreements with a provider.**

   Because cloud computing is designed for the sharing and on demand of resources, the allocation of these resources becomes extremely important. Both providers and users of cloud services need to define the exact terms of use and service. In case of an organisation using external cloud providers, this takes the form of a Service Level Agreement (SLA). Even in case of private clouds, some kind of policy for service delivery will exist, and whereas it may not be as formal as a SLA, it should cover similar territory. This means is that customers are able to tailor service offerings to specific needs, provided these are offered by the Service Level Provider (SLP).

In 2011, the American National Institute of Standards and Technology's (NIST) published their definition of cloud computing (Mell & Grance 2009). This definition has now become the de-facto definition for cloud computing (at the time of writing, Google Scholar approximates the number of citations at 1138). To ensure clarity of concepts and vocabulary, the NIST definition will be used for the remainder of this thesis.

2.2 Cloud Service Models

A common way of distinguishing between different cloud services is by dividing them into layers, such as Software as a Service (SaaS), Platform as a Service

(PaaS) and Infrastructure as a Service (IaaS) (Lenk et al. 2009). In a layered model, each layer builds on services offered by the layer below, and in turn offers services to the layer above. Each layer uses its own information types (data classes and properties) to provide specific functionality.

NIST provides the following definitions of the services provided by each layer:

- **Software as a Service (SaaS).**

The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user specific application configuration settings.

- **Platform as a Service (PaaS).**

The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

- **Infrastructure as a Service (IaaS).**

The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls).

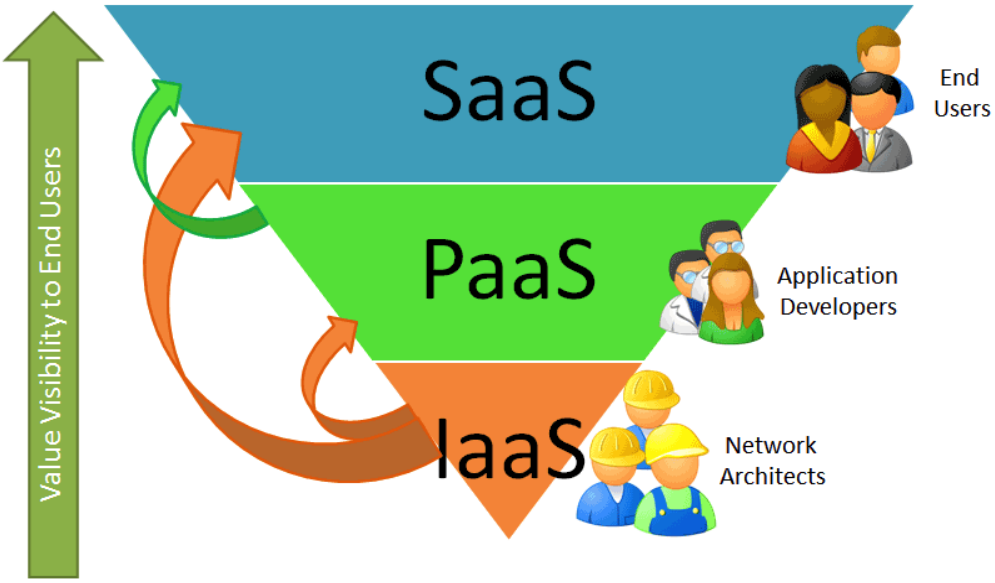

Figure 1 - The three cloud service levels as visualised by SaaSblogs. (Schuller 2008)

2.3 Types of Cloud Offerings

So far, this thesis has distinguished between cloud providers and cloud users, however, in reality the distinction is not so clear. Cloud services can be offered according to a number of so-called deployment models, depending on the relationships and types of providers and users. The most commonly used deployment models are:

1. **Private clouds.**

Here, the cloud services are offered for the benefit of a single organisation with multiple users. This type of solution is usually chosen by large organisations, with high or specific requirements for security or computing power.

2. **Public clouds.**

Here, the cloud services are open for use by the general public. This type of cloud can be owned by different types of organisations with multiple different users.

3. **Hybrid clouds.**

Here, the cloud infrastructure is a combination of a number of different cloud

infrastructures, such as private and public clouds.

4. **Community clouds.**

Here, the cloud infrastructure is provided for the benefit of a community of users that share the same technology needs. For example, a community cloud could be provided for a specific scientific community.

In the research presented in this thesis, no specific deployment model is implied unless otherwise specified (Badger et al. 2012).

3 Related Research

3.1 Previous Research

Content Management Systems (CMS) are widely used for organisations to publish information, to keep transactions and records, and so on. By this wide acceptance of electronic documents and records, organisations are facing demands for the safe archiving of electronic records in their repositories. However, in general, CMS in use today do not offer the required level of functionality for an organisation that has a responsibility to maintain its records. It therefore becomes necessary to transfer the records to be retained to a Records Management System (RMS).

CMS and RMS are seldom interoperable out of the box, making archiving of retained records difficult. There are many reasons for this interoperability: Differences in the used metadata schemes, lack of metadata conversion and incompatible export/import processes. Up to now, the solution to these problems has been to add records to the archive by hand, or to create custom programs for records transfer, made to match the existing software and hardware profile. Neither of the above solutions is optimal. In his previous research, the author, the author proposed a lightweight approach to the problem of integrating Content Management and Records Management software. The approach was based on a three layered model for the organisation of a corporate records management system. The model allowed for the connection of one or more CMS to a RMS by making it possible to automatically transfer and ingest retained records for archival.

Using the proposed model, the author developed a system named ATLAS (Automated Transfer Lightweight Archive System). ATLAS was designed to connect multiple CMS with different metadata schemes to a single records repository, enabling automatic archiving of records submitted by users. Each CMS is registered in ATLAS, along with a metadata crosswalk that translates CMS metadata into a metadata format that can be imported into the RMS. This means that CMS metadata terms that have an equivalent target in the RMS metadata scheme were automatically reused.

ATLAS also supports registration of additional CMS by allowing administrators

to upload metadata crosswalks in XML/OWL. XML/OWL was chosen because it provides organisations with a common vocabulary for metadata terms, including the relationship between these terms. This makes searching the crosswalks easier, since the location of the terms and their interrelationship is defined in an ontology.

ATLAS uses RSS 2.0 as a protocol for transferring records and metadata. Because it uses open protocols and technologies, such as RSS and XML, ATLAS is designed to work with existing organisational CMS and RMS. It also makes it possible for organisations to use existing tools to expand the functionality of ATLAS by adding support for technologies such as authorisation and track-back.

With this research project, the author demonstrated that the cost of Records Submission in an organisational setting can be significantly reduced by using a system built using the three layered model, exemplified by a system like ATLAS, constructed entirely with off the shelf tools and open technologies.

The fact that systems implemented using the three layered model do not require any significant reprogramming of the organisations CMS or RMS for records and metadata transfer, coupled with the fact that it is not tied to any one software solution, makes it easy to implement in an organisations existing technological environment. The ATLAS archiving system constructed for this research was able to transfer content and metadata successfully from an out of the box CMS to a records repository built on Dspace. The ATLAS solution was shown to be significantly less costly than a manual export/import process, and more flexible than a solution based on CMS specific plug-ins or ad-hoc export/import scripts.

The system presented in this thesis builds on many of the technologies explored in the previous research. In particular, the idea that there are benefits in abstracting functionality and standardising services in the design of complex systems. Another recurring theme is that it is possible to automate most of the records submission and ingest process, especially when it comes to the creation of preservation metadata (Askhoj et al. 2007).

3.2 Current Research Related to Cloud Archival

In recent years, cloud computing have become the subject of a wide range of research initiatives, and this trend appears to be growing. In 2012, IEEE listed no less than 3,147 individual articles with the keyword “cloud computing”. This amount of activity is likely due to the scope of cloud computing, covering many aspects of computer science and related fields, including security, cryptography, the management of large datasets, etc. This section section will present a number of related current and recent initiatives dealing with the use of cloud computing for archival purposes and the preservation of data in the cloud. This section presents research related to four different aspects of cloud computing: Digital archive systems using cloud computing, Dependable / persistent storage, cloud interface standardisation and metadata.

3.2.1 Digital Archive Systems Using Cloud Computing

As stated earlier, the cloud has not been used extensively for the creation of archives, due to the newness of the technology, the lack of standards and the problems with guaranteeing long-term preservation. However, the paper “A medical image archive solution in the cloud” shows the feasibility of creating such a system in practice. In the paper, the researchers build a prototype imaging archive system using the Microsoft Windows Azure cloud computing platform. The system is comprised of three parts, a Digital Imaging and Communications in Medicine (DICOM) server for store/query/retrieve requests; a DICOM image indexer that parses the metadata and stores it in a SQL Azure database; and finally a web UI for searching and viewing archived images based on patient and image attributes. Unfortunately, the paper doesn't address the problems of long-term preservation, but it has nevertheless been included here as an example of a proof of concept (Chia-Chi Teng et al. 2010).

A theoretic, but far more comprehensive description of a digital cloud archive is presented by Quyen L. Nguyen and Allan Lake in the paper “Content Server System Architecture for Providing Differentiated Levels of Service in a Digital Preservation Cloud”. Here, a Content Server System Architecture is proposed to create a Digital Preservation Cloud. The architecture forms the core of a so called Long-Term Digital Preservation as a Service (LDPaaS), designed to take the burden of preservation away from the content producers. Central for the

system presented is the concept of delivering differentiated levels of preservation services. For example, a low level of service would only provide bit-level preservation, whereas a high level will capture not only text, but also any accompanying formatting. Other digital archiving services are also available as a service (namely Ingest, Discovery, Access, Storage and Content Server). The paper also presents a cost model for the LDPaaS, based on the volume of digital records, and the level of service.

One interesting aspect of the paper by Nguyen and Lake is the introduction of a layered model, somewhat similar to one presented by the author of this thesis a year before Nguyen and Lake. However, Nguyen and Lake and Lake have opted for using a more system oriented way of creating layers, dividing their architecture into the following 4 layers:

- Physical Layer – Storage of metadata and Digital Objects
- Service Layer – Common services, logging, transaction management/ Authorisation and search engines with possible adaptors
- Process Layer – Index Manager, Search Manager, Metadata Manager, Digital Object Manager and Configuration Manager
- Interface Layer – RESTful Service Manager, SOAP Service Manager

Nguyen and Lakes research is in many ways similar to parts of the research presented in this thesis. The objective, to create a cloud based digital archive with functionality provided as abstracted services, is the same. However, the research presented in their paper is focused on the provision of different levels of service and calculation of the cost associated. The paper was presented at the 2011 IEEE International Conference on Cloud Computing, and is still ongoing. It will be interesting to see an actual system built on the model proposed (Nguyen & Lake 2011).

3.2.2 Dependable/Persistent Storage

An important prerequisite for the research presented in this thesis is trusted storage. If bit-level integrity of cloud data cannot be guaranteed, it will make for a poor digital archive. The best known organisation offering cloud storage for the express purposes of archiving is the non-profit organisation DuraSpace with

their DuraCloud offering. DuraCloud provides redundant storage of data spread over one or more underlying of cloud providers, accessible via a standard API. Furthermore, DuraCloud is available as open source, and has a close relationship with the two biggest open source digital repository software projects Dspace and Fedora. DuraCloud allows replication of content to multiple providers and locations, synchronisation functionality and rudimentary data integrity checking by checking content ID and associated checksums using MD5.

That data is stored in a redundant and checkable manner, coupled with the fact that the DuraCloud application is open source are compelling reasons for an organisation looking for a storage solution.

Most providers of cloud storage will guarantee the integrity of any data they manage to a high specification. However, there are still a few drawbacks compared to traditional data centres. The fact that the data is not in-house can make auditing difficult, and in some cases, a customer may be forced to take the word of the provider that their data is safe. Indeed, unscrupulous providers may even attempt to hide a loss of data integrity to protect their own reputation. Furthermore, with a cloud solution, it is not possible to adopt traditional cryptographic primitives for the purpose of data integrity protection because this may conflict with the way data is stored by the provider. These problems are dealt with in the paper “Toward Secure and Dependable Storage Services in Cloud Computing”. Here, Cong Wang et al. present a method for on-demand data correctness verification that can be initiated by the cloud users themselves. This not only provides an independent and advanced alternative to any integrity check offered by the cloud provider, it also supports auditing carried out by a third-party (such as an independent auditing company specialising in data integrity). Perhaps the most impressive part of the proposed solution is that it can be conducted without explicit knowledge of whole data files and that it can be carried out dynamically. In other words, the verification can be carried out on live datasets that have been updated by users while still offering error localisation (misbehaving server identification) in case of attack or failure. Based on tests with real datasets, the method proposed by Cong Wang et al. is proven to be resilient to media failure, and malicious data modification attacks (Cong Wang et al. 2012).

3.2.3 Cloud Interface Standardisation

A problem often raised in cloud computing is the lack of API and management interface standardisation. Without such standardisation it is difficult to install and manage cloud systems that span several different providers and migrate existing solutions (Yamato et al. 2012) (Leavitt 2010).

This problem is the object of the Cloud Data Management Interface Reference Implementation (CDMI) currently developed by The Storage Networking Industry Association (SNIA). SNIA is a non-profit standard organisation, counting among its members some of the largest tech companies, such as Cisco Systems, Oracle and Hitachi Data Systems. CDMI is a data management interface for Cloud Storage. It allows administrators to use a standard interface to manage and perform create, read, update and delete (CRUD) operations on stored data across different clients. CDMI commands are sent using standard HTTP operations built on top of JSON[1] using a RESTful[2] interface where possible. Apart from defining the defines the functional interface that applications use for operations, CDMI also allows for data and data containers to be tagged with special metadata. This metadata is used to tell a provider what services are needed for the data, and can be used for requesting services such as scheduled backups, special retention requirements, encryption etc. Because this metadata is standardised, it can be used across all providers that support CDMI. Finally, because not all providers may support the entire CDMI specification, the interface can be used to query what the capabilities of a specific cloud storage offering is in advance.

CDMI provides a number of clear benefits to a digital archive implementation. An archive may not want to “put all its eggs in one basket” (i.e. rely on a single cloud storage provider) so having a standardised interface simplifies management and means that administrators only need to use one set of commands. The ability to set metadata on containers and their contained data elements can help with finding these by performing a queries for specific metadata values. The fact that the metadata is standardised and independent of

1 JSON (JavaScript Object Notation) is a simple data-interchange format.

2 REST is an architectural style for distributed resources designed by Roy Fielding. It uses simple HTTP to communicate between machines. It is characterised by the following 6 architectural traits: Being Client–server, Stateless, Cacheable, Layered, having Code on demand (optional) and a Uniform interface.

specific providers makes it much easier to migrate data from one cloud vendor to another (Storage Networking Industry Association 2012).

3.2.4 Metadata

Later parts of this thesis will discuss various applications of preservation metadata to a digital cloud archive. However, there is very little research to be found dealing specifically with the implementation archival metadata sets in the cloud (it should be mentioned that some projects may incorporate individual pieces of archival metadata, for example CDMI makes it possible to set a retention expiry flag on a particular piece of data or container). One of the few examples available deal with the capturing of provenance metadata. In the paper Provenance for the Cloud, Kiran-Kumar Muniswamy-Reddy et al. present three alternative protocols for storing provenance using cloud services. Before proposing the protocols, the paper defines four critical properties of provenance. These are:

1. **Provenance data-coupling.**

   This states that system records describing data and provenance must match. In other words, the provenance must accurately describe the data recorded.

2. **Multi-object causal ordering.**

   This states that ancestors described in an object's provenance must exist, so there are no dangling provenance pointers.

3. **Data-independent persistence.**

   This states that provenance must persist even after the object it describes is removed.

4. **Efficient query.**

   This states that the system must support queries on provenance across multiple objects.

As can be seen by the critical properties, the purpose of the paper is to capture cloud provenance as a means of verifying data authenticity and identity. The paper uses a specially developed Provenance Aware Storage System (PASS) to record the attributes: command line argument, environment, variables,

process name, process id, execution start time, the file being executed, and a reference to the parent of the process. This metadata is then stored in the cloud using one of the proposed protocols. After evaluating the protocols, the paper concludes that their implementation is feasible using off-the shelf cloud components, and that although there are challenges to overcome (cost, performance, security etc.) these are not insurmountable (Muniswamy-Reddy et al. 2010).

Even though the PASS and protocols designed are designed for data authentication purposes, they could add a lot of value to a digital archive. Not only to validate the integrity of Digital Objects, but also to populate preservation metadata. An archive must be able to rigorously track events that occur over the course of the Digital Object's life cycle, and the metadata captured by PASS would be of great value.

Finally, there is another promising project underway to solve the problem of inter-repository exchange of preservation metadata: Towards Interoperable Preservation Repositories (TIPR). Here, Caplan et al. have defined a new package format for exchanging information and API across distributed preservation repositories (Caplan et al. 2010). Whereas, this research is currently focused on metadata that has already been ingested into repositories, the ability to exchange API information would be useful in the cloud, where differences in APIs are a big barrier to data exchange.

3.3 Current developments related to the OAIS Model

The Open Archival Information System Reference Model (OAIS) will be presented in more detail in section 6 of this thesis. However, in 2012 a new version was published. Because this model forms such an important part of the work presented here and for digital archiving in general, it would be advantageous to present some of the changes and their implications for the research presented here.

The new version of OAIS was released by CCSDS on the14th of June 2012. This is the first major update since the initial release back in 2001. It is a testament to the success of the original version that it is still the de-facto standard for digital archives more than 10 years after its initial release. The

changes in the model are evolutionary rather then revolutionary and introduce revisions to the descriptions of some functional entities without changing the overall framework (Sierman 2012).

Some of the more noteworthy changes that could impact this thesis are:

- Preservation Description Information (PDI) now contain an element for Access Rights information. This is a valuable change as Access Rights in the old version mainly dealt with Consumer Access Rights, which is problematic because there may be Access Rights limiting what an archive can do with a digital object and this should be reflected in the PDI.
- There is a new definition for “Other Representation Information”, which is described as *“Representation Information which cannot easily be classified as Semantic or Structural. For example software, algorithms, encryption, written instructions and many other things may be needed to understand the Content Data Object...”*. This is potentially interesting because it opens the door for cloud system related metadata (which is neither semantic or structural) to be described as part of Representation Information.
- Perhaps the biggest change is the introduction of the new concept Authenticity. This concept is being introduced in relation to transformation of digital objects and preservation metadata. Transformation was already present in the old version of OAIS, but there was no way to define certain elements or properties necessary for guaranteeing that the result of the transformation was authentic. This is now possible by defining Transformational Information Properties. These are key properties that must be kept after the transformation, chosen for their ability to demonstrate authenticity.

There are other changes as well, but these are relatively minor with regards to the subject matter presented here. If there is a specific need to distinguish between versions of OAIS in this thesis, this will be done in the text. Finally, it should be noted that like version one, the new version of OAIS does not address cloud systems.

4 Research Problem

4.1 Research Background

As stated in the previous section, trying to solve all problems relating to the storage of records using cloud computing is unrealistic. This thesis is focused on creating a model for an archiving system using cloud computing that can be used by businesses and organisations to ensure the long-term preservation of digital contents. In other words, the object is not only to create a digital archive in the cloud, but to deliver a solution whereby an organisation can easily transfer their archive-worthy records into the cloud, while still making sure that those contents can be accessed and used in future. Such as solution is necessary because of the difficulties in ensuring preservation when both the Digital Objects themselves as well as the technological infrastructure are managed by a third party.

Common for cloud computing services these services is that responsibility has been outsourced to a third party, specialising in managing large scale data centres and data sets. With the increasing adoption of cloud technology, great progress has been made in reliability, backup and data protection. However, even assuming service providers are able to guarantee safe bit-level storage of this data, this does not cover other aspects of preservation, such as renderability, understandability, authenticity. Providing metadata to support these activities also becomes important (Sugimoto 2007).

This transfer of service and responsibility to one or more third parties can make it hard to guarantee the reliable storage and preservation of records. Changes in available services, difficulty in emulation and migration, inadequate preservation and so on may result in the loss of information (Jain & Bhardwaj 2010).

Existing archive models, such as OAIS (the Open Archival Information System Reference Model developed by the Consultative Committee for Space Data Systems is an extensively used reference model for archiving systems) have been established specifically to deal with archiving and preservation, but they may be inadequate or hard to apply when it comes to the cloud. It is necessary to discover now if existing archive models are applicable in a cloud computing

environment or if new solutions are needed before it is too late (CCSDS Secretariat 2002).

4.2 Requirements Analysis for a Cloud Archiving System

Before defining the requirements of a cloud archiving system, it it worth taking the time to examine the purpose of records management and digital archiving.

There is no single, worldwide accepted definition of an archive. But the American National Archives and Records Administration provide the following explanation of an archive:

"The noncurrent records of an organization or institution preserved because of their continuing value." (Daniels 1984)

The above definition can be applied to the archives described in this paper, with the addition that the records to be stored are in digital rather than paper (or other physical) format. Unfortunately, the same lack of a definitive taxonomy of terms is to be found when it comes to digital archives. In this context, digital archiving should be understood to include all the actions required to maintain long-term access to digital records beyond the limits of storage media failure or changes in technology. When using the term preservation, the meaning is not solely the safe storage, but also the process of keeping records accessible, searchable, and usable over time. The records may be "born-digital" materials created for a specific purpose, or the products of digitisation projects. This can include many kinds of recorded information, regardless of media or characteristics, made or received by an institution or organisation (Beagrie & Jones 2008).

Similar to archives, regulated companies and organisations need to manage their digital records. Any document in any format that is evidence of a business transaction is a record and needs to be managed. Records Management is the systematic control of records throughout their life cycle. Or as ISO 15489:2001 puts it:

"The field of management responsible for the efficient and systematic control of the creation, receipt, maintenance, use and disposition of records, including the processes for capturing and maintaining evidence of and information about business activities and transactions in the form of records". (ISO 2001)

This means that organisations must put in place systems and processes that makes this control possible, from the creation or accession of a record until such time at it is destroyed. There are two overall reasons why records management is necessary. The first one is compliance. Organisations today need to comply with a large number of rules and regulations when it comes to records. Failing to comply with these rules can lead to substantial penalties. The other reason for records management has to do with the benefits of having access to and control over business records. Examples include: Less time spent looking for records, storage cost reductions, the ability to re-start business in case of disaster, the ability to do knowledge management/knowledge sharing, etc. In order to address the challenges mentioned above, most large organisations have implemented systems that have functionality to manage records. This can either be functionality added to existing systems, such as databases, content management systems, finance systems and the like, or it can be dedicated Enterprise Records Management Systems (ERMS) / Electronic Document and Records Management System (EDRMS). Functionality offered by such systems include classification, business retention schedules, physical file tracking, automated destruction etc. Common for all such systems is that they store records in a very controlled environment (McLeod 2002).

One of the important differences between records management in organisations and institutions and digital archiving is that for an archive, preservation and other archiving tasks are the core function, whereas records management is just one of a number of necessary processes needed to ensure the smooth running of an organisation. This means that there are often few resources available in organisations to manage records, hire specialised staff etc. This is especially true of smaller organisations and those in areas that are not subject to strict regulatory oversight. This is the reasons why many organisations, from private companies to government departments chose to transfer their records to a dedicated archive (Robles & Langemo 1999).

4.3 Challenges in the Use of Cloud Computing for Electronic Records

One of the reasons for the amount of research in the use of cloud computing for

electronic records is the large number of challenges when it comes to long-term preservation. Whereas paper records may suffer from deterioration over time, people are still able to display and read paper records created fifty or a hundred years ago. Not so for digital records, where different storage media and different versions of software can render even recently created documents unreadable.

The problems relating to the preservation of electronic documents are well known, and steps can be taken to ensure systems can provide long-term accessibility and readability of electronic records. However, with cloud computing, the management and responsibility of infrastructure, systems and data may no longer reside in the organisation in which the electronic records are created or in the archive where they are managed (Huth & Cebula 2011). This transfer of service and responsibility to one or more third parties can make it hard to guarantee the reliable storage and preservation of records. Changes in available services, difficulty in emulation and migration, inadequate preservation and so on may result in the loss of information. From a preservation perspective, one of the biggest disasters that can occur is a complete or partial loss of data (including the loss of the metadata that allows us to read and understand the preserved data). This worst-case scenario happened in 1998 where the online storage service MediaMax lost 45 percent of their customer data due to a system administration error (Krigsman 2008). However, partial outages have been known to occur even with global leaders in cloud computing such as Google and Amazon. There are a number of scenarios where such a loss can occur, such as the cloud provider going out of business, being hacked, or suffering from software, hardware or human error.

The following list summarises some of the main challenges as reported by a number of advisory bodies. For reasons of clarity, the challenges have been split into categories.

1. **Security concerns**

Whereas security is not directly a preservation concern, it is among the top concern of organisations wanting to implement cloud computing (Cachin et al. 2009). In the same way, if a storage solution is insecure, it is not suitable as a repository. The fact that clouds are distributed and internet based means that there are many factors to be considered when formulating an overall security

strategy. When it comes to records management, the priority is on ensuring the security and confidentiality of the actual records stored in the system. These need to be managed in a secure environment, with adequate or specific standards compliant systems in place, e.g. access control, encrypted access, backup etc. Current cloud providers already provide varying degrees of security measures in their offerings, however some organisations may be subject to additional requirements for records management such as Department of Defence (DoD 2007) or VERS Standard compliance (PROV 2003). Providing such compliance and auditability lies outside the scope of many cloud providers (although things are improving as offered services mature) Such security requirements provide another barrier to cloud adoption. There are also requirements for auditability in legislation such as the American Sarbanes-Oxley act. Any cloud solution that needs to be compliant must have functionality for capturing and storing audit-trails built in (Securities and Exchange Commission 2003).

2. **Privacy concerns**

With data stored by an external provider, there is always the possibility that it may be accessed and read by a third party. Privacy breaches can happen as the result of malicious hacking, but also as part of the provider willingly handing over data to private parties for commercial gain or to government or law enforcement agencies. An example of the former is companies like Facebook using personal information for private gain. As an example of the latter, many Asia Pacific countries do not have comprehensive national data protection laws. Even storing data in a western democracy like the US is no guarantee of privacy, as shown by the Patriot Act that gives a law enforcement agencies the right to access privately hosted data without a court order (Office of the Victorian Privacy Commissioner 2011) (Story & Stone 2007).

3. **Vendor specific concerns**

Relying on one particular cloud provider carries with it a number of risks. an organisation may wish to change providers or move data back in-house, e.g. due to increases in pricing. Organisations that for some reason want to move their data out of the cloud are faced with a number of choices. They can either

attempt to export the data and accompanying metadata itself, or they can try to migrate parts (such as a specific database or software application) of the cloud to another providers infrastructure. Both of these approaches are problematic. In the first case, whereas an export of data and metadata from a cloud based application might be possible, obtaining a sufficient amount of administrative and structural metadata in the right format is another matter. And even if all necessary data was extracted, the organisation is still faced with the challenge of migrating this data to a new system - a system that may have different requirements for formatting, metadata and usage. In the second case, instead of exporting data out of the application where it is kept, a less costly option would be to simply migrate the parts of the cloud that hold the data to a different infrastructure. This, however, can also be challenging. Because of the interrelated nature of cloud computing, one part such as an application or service may be reliant on specialised functionality or services provided by another part. Such interrelation is common in complex software environments, but is aggravated by a lack of standards and use of proprietary solutions in current cloud systems (Khajeh-Hosseini et al. 2010).

4. Technical challenges

Currently, planning for long-time preservation of data in the cloud is very difficult as it needs to take into account the ever changing architecture of cloud providers. As mentioned above, migration planning becomes difficult when organisations outsource responsibility for records to a third party over whose systems they have only limited influence. That Content Management Systems (CMS) used for document creation have limited long-term preservation functionality is not a new problem. A popular method of solving this is the transfer of records to a records repository with proper long-term retention management, as exemplified by the OAIS model. In a cloud environment this is also possible, provided the CMS in question lives up to the requirements for an OAIS, for example, it must have the ability to provide properly formatted Submission Information Packages (SIP) and that the transmission must take place in such a way that data security is guaranteed. Another thing to keep in mind is that whenever records and metadata is exported out of one system and ingested into another, there is an inherent loss of data from the originating

system, due to differences in metadata schemas, document conversion etc. Apart from changes to cloud infrastructure, another problem is the ability to insure data integrity and authenticity. One of the benefits of cloud computing is that organisations do not need to know the specifics of how cloud provider IT systems function, however this very same “black box” approach makes it almost impossible to know whether data integrity has been maintained before it is too late (Metsch 2010).

5. Legal considerations

As mentioned under Privacy Concerns, some types of information may have restrictions on them that prohibit them from being stored the cloud. These restrictions can be either legislative or decided by organisational policy. As an example of a legislative restriction, European Data Protection Law states that records may not be moved to countries that do not have data protection laws with protections similar to those in the country where the records were originally maintained (European Parliament, Council 1995). This can be difficult to guarantee because many cloud providers store customer data in data centres that are located in different geographic locations. This means that some organisations have chosen not to trust the cloud with any of their vital records as a matter of policy. It should be said that in recent years, a number of cloud providers have started to offer data storage based on geographic location.

Another concern when it comes to protecting privacy is the difficulty in guaranteeing that data stored with an external party has been securely deleted. Most organisations operate under strict Retention and Disposal Authorities, that state the period after which records can be destroyed. Destruction in this case means complete and irreversible destruction, including any back up tapes and decommissioned discs holding data. Because cloud data is often hosted on many servers in many locations, it may be difficult to know whether it has been securely destroyed. (National Archives of Australia 2011)

The purpose of this list of challenges is not to expand the scope of research, but only to highlight the fact that when dealing with electronic contents in the cloud, there are a multitude of factors to take into consideration, both directly and indirectly related to secure storage and preservation. This is one of the main reasons why many producers still need to submit their digital objects to

traditional digital archives. However, based on the massive growth in cloud computing, these challenges do not seem to be deterring many companies and organisations eager to benefit from the advantages of cloud computing (Leavitt 2010).

4.3 Research Problems

As can be seen, there are a many challenges to address. As it is unrealistic to deliver a comprehensive answer to all of these, this thesis attempts to answer the following questions: To what extent is it possible to apply existing models for digital archives to a cloud environment, as exemplified by the OAIS model? If there are problems in this application, how can a new model be developed to integrate the requirements of both producer and digital archive when building a cloud-based digital archive? What would such a system look like, and how can it be implemented? Finally, how can such a system be evaluated?

The thesis is organised as follows: Section 2 provides a definition of Cloud Computing and examines some of the characteristics that comprise a cloud system. Section 3 looks at current research related to archiving of cloud contents, focusing on digital archive systems using cloud computing, dependable/persistent storage, cloud interface standardisation and metadata. Section 4 presents the thesis research problems, followed in section 5 by the research method. Section 6 introduces the Open Archival Information System (OAIS) Reference Model and looks at the problems in applying this model to a cloud environment. Section 7 presents a new layered model to address the problems of archiving in the cloud, explaining why such a model is necessary and how if works. In chapter 8 is a theoretical case study for how the model could be applied in real-life, using the transfer of records from the Japanese Government to the National Archives of Japan as an example. Sector 9 presents an application profile for cloud archiving systems built on the entities and functionality of the layered model. Section 10 presents an ontology for preserving digital content in the cloud using PREMIS preservation metadata and written in OWL. Section 11-12 explains how the ontology may be used for the validation of information packages in a test scenario by querying preservation metadata in RDF with SPARQL. The thesis ends with a discussion of how the ontology may be used and a conclusion in section 13 and 14

respectively.

5 Research Method

The starting hypothesis of this thesis is that computing represents a fundamental shift in the computing landscape and it is not covered sufficiently by traditional methods of digital archiving, as represented by the OAIS model. The first step in the research should therefore be to verify whether this hypothesis is true, and if so to explore what parts are not well covered, and what is needed to overcome this discrepancy. As stated previously, the object of this research is to create a practical system that integrates the requirements of both producer and digital archive. With that in mind, it is necessary not only to look at how OAIS may or may not be suitable as a model. Cloud computing does not only create problems, but also opportunities. For example, cloud computing is, that services can be easily shared between a number of systems. As mentioned earlier, organisations can select whatever level of services they require and build on these to create the systems they want. Because of the scalability and networked nature of cloud computing, it is possible for systems to share a common execution environment, or a common storage solution. For example, the records producing institution may share the same storage solution as the archive it is submitting records to. Such benefits should be taken into account when evaluating a model.

Based on the outcomes of the above, the next step is to develop a simple model for the creation of a digital archive system in the cloud, describing the major functional entities and their relation to each other. Again, this model should be evaluated against the questions raised under the Research Problems and once complete, it should be evaluated by examining whether it is applicable in the real world.

As stated, the ultimate goal of this research is to develop a model that can be applied by both producers and archives when using cloud based systems. In order for this to be possible, it is necessary to define rules for interoperability and assign responsibilities. To achieve the former, an application profile will be designed to guide system implementors in their choice of metadata and to promote the linking of data within different communities. In combination with the application profile, plans for implementation will be developed, and whereas it may not be possible to build a complete cloud archive system, it is hoped that it

will be possible to develop enough of a framework to perform an evaluation and form some conclusions regarding the applicability of the proposed solution.

6 The OAIS Reference Model

Traditional digital archives receive their records and accompanying metadata from a producing organisation or institution in a predetermined package format. The archive and producer agree on a set of requirements for submission, decided by the archive in order to ensure an easy ingest for the archive. Once contents have been ingested, they are stored and managed in a data centre, where the archive complete control of the technological infrastructure and Digital Objects. Using this infrastructure and digital archiving software, the archive provides access to archive users, while ensuring the ongoing preservation of the archiving collection. Examples of archives that operate using this model include the Japanese National Archives, The National Archives of Australia and the Public Records Office of Victoria.

This model is similar to the one presented in the Open Archival Information System (OAIS) Reference Model (ISO 14721) developed by the Consultative Committee for Space Data Systems, released in 2001. Not only is OAIS the *de facto* standard in digital preservation, as a framework it also covers the same overall functionality of a digital archive, whether it is hosted in the cloud or otherwise (CCSDS Secretariat 2002).

In other words, The OAIS serves not only to explain the structure of an archive and its functional entities; it also provides models and concepts for the information to be preserved. The overall framework and major information flows of the OAIS can be seen in figure 2.

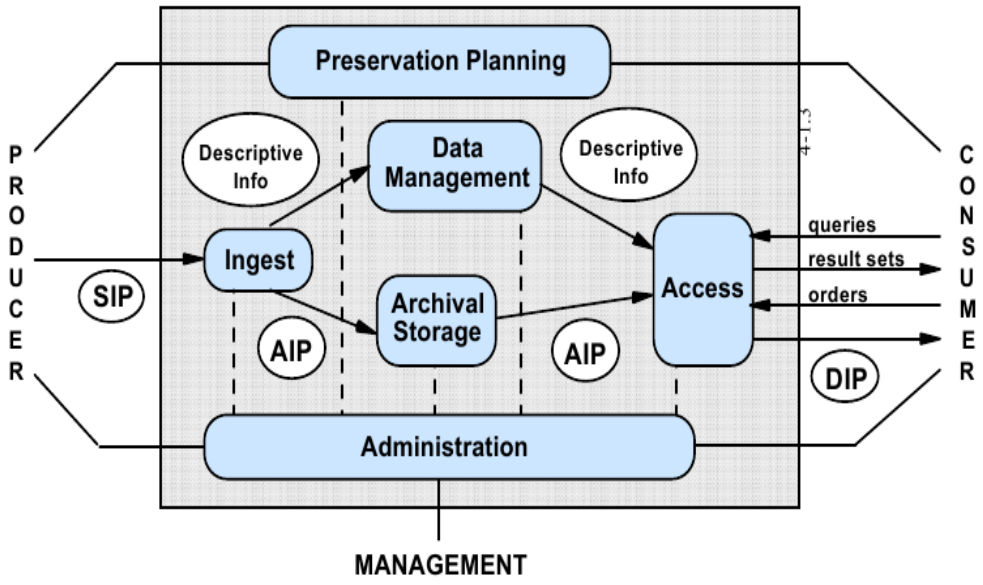

Figure 2. OAIS Functional Entities

6.1 Functional Elements of the OAIS Model

The OAIS archive represented in the centre of figure 2 as a grey box consists of a number of entities: Ingest, Data management, Archival Storage, Access and Administration. Each of these entities performs specific functions, such as accepting submissions, planning preservation and administrating the archive.

The main stakeholders outside the OAIS are Producer, Management and Consumer. The Producer submits records to the archive in an agreed format, known as Submission Information Packages (SIP). These packages are imported into the archive by the Ingest entity that performs quality assurance on the submitted packages. Based on the SIP, Ingest generates an Archive Package known as an Archival Information Package (AIP) for storage in the archive and sends updates to the Archival Storage and Data Management entities. AIPs are stored in Archival Storage and their descriptive information and administrative data is handled by Data Management. Consumers can obtain records from the archive by querying and ordering from the Access entity. Resources are delivered to Consumers in the form of Dissemination Information Packages (DIP).

Since this thesis covers not only historic records, but also records produced in the course of business, the word "record" has been used to describe the objects to be archived. This is in accordance with the ISO 15489: 2001 standard that

defines a record as: *"information created, received, and maintained as evidence and information by an organisation or person, in pursuance of legal obligations or in the transaction of business*" (ISO 2001).

In the OAIS Model, the object to be archived is known as a Data Object. A Data object can be either a Physical Object or a Digital Object such as a file or a database entry. In order to ensure that the Data Object is understandable to its target audience, it needs to be stored with Representation Information that maps it to more meaningful concepts. A Data Object with accompanying Representation Information is known as an Information Object.

In an OAIS, Information Objects are associated with additional information to ensure their correct transfer and storage. This is referred to as an Information Package. An Information Package is a conceptual container containing the Content Information and Preservation Description Information (PDI). The Information Package is defined by Packaging Information relating the package components, for example directory structures or ZIP files.

6.2 Problems Applying the OAIS Model to a Cloud Environment

It would be ideal if the OAIS model with its Functional Entities could be applied directly to a cloud environment, where services can be shared and abstracted in layers and where services such as storage and data management can be outsourced to a third-party and paid for on-the-fly. There are, however, some areas in the OAIS that make integration with cloud computing difficult.

1. The fact that the functional entities in OAIS are interdependent, makes it is difficult to transfer responsibility for parts of an OAIS archive to an external service provider. In other words, it may be difficult to get an external party to provide the exact functionality specified by OAIS for a functional entity or to make sure that any functionality not provided by the external provider is covered elsewhere. For example, if an organisation is looking for a storage solution offering bit-level integrity for Digital Objects to use as a back-end for an archiving system, this would involve overlapping functionality from the Management, Data Archiving and Archival Storage entities.
2. In OAIS, the burden of creating Submission Information Packages (SIP)

is left to producers, who must meet the requirements of the OAIS archive. This task can be very resource intensive, depending on the strictness of the requirements and how many archives the producer submits to. This can lead to producers holding on to records for long periods, before submitting them in bulk. This can significantly delay preservation planning. With cloud computing and the shared platform it offers, Digital Objects can be made accessible to an archive without delay, allowing early preservation planning.

3. With cloud computing, the need to include Digital Objects and metadata in Information Packages disappears. With a shared, trusted platform, producers only need to provide the information (URI or similar) of where the Digital Objects are stored. However, the OAIS Model does not specify the requirements and functionality of such a shared platform.
4. The OAIS doesn't cover the initial stages of the Document Lifecycle (the Create, Use, Manage stages). It can be argued that these stages lie outside the scope of an archive. However, the nature of the events in these stages and how well they are documented can have a huge impact on how easy it will be to carry out preservation work later on. In other words, a model that ensures good interoperability would not only decrease cost, but could also help with the creation of preservation metadata.

7 A Layered Model for Cloud Archiving Systems

7.1 The Need for a New Model

It is equally possible to apply a layered model to archiving systems. This can be illustrated by a simple two-layered model for an archive framework (figure 3). The bottom "Platform Layer" represents a trusted digital cloud repository offering guaranteed document integrity at the bit-level. The top "Interaction Layer" contains applications that can access documents stored in the below layer. In this scenario, Producer applications can save documents directly to the repository, trusting that secure storage is provided. (Sugimoto 2007).

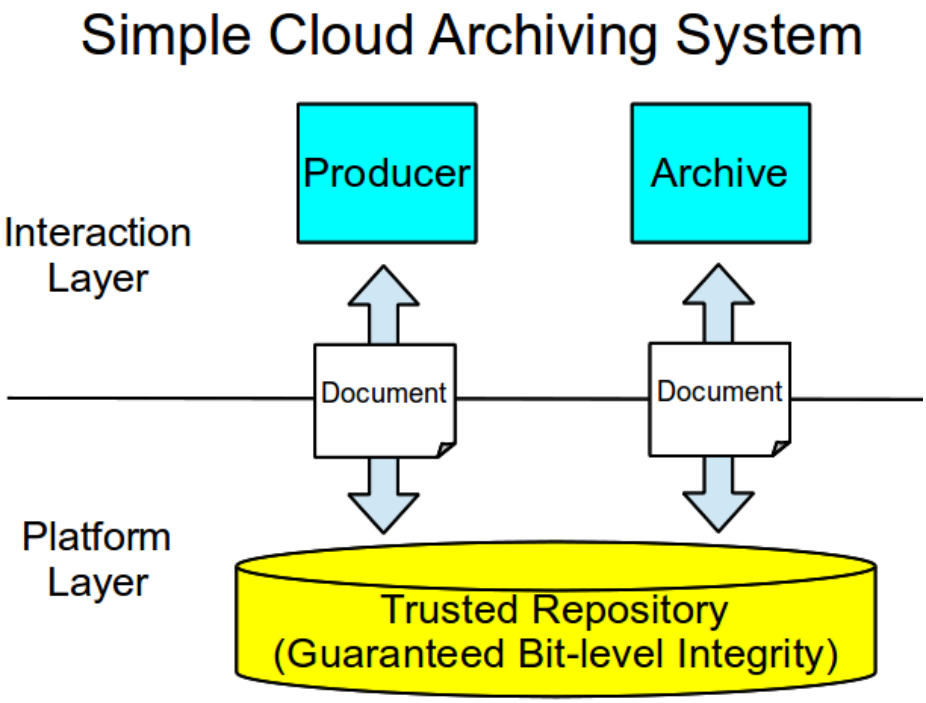

Figure 3. Simple model for a cloud archive

Because of their scalability and networked nature, cloud services can be easily shared between a number of systems. For example, the contents producing institution may share storage with the archive it is submitting records to. These documents would be immediately available to an archive. This would bring benefits to both parties by reducing the need to duplicate services.

7.2 Benefits of a Layered Model

As can be seen from Figure 3, a layered model was chosen to form the basis of research. In computer science, an abstraction level is a generalisation of a model that does not rely on any one specific implementation, ans such models are often used in system architecture to create a logical division of services. The benefit of a layered model is that once data types and services are defined, layers can be abstracted. This means that when considering any specific layer it is possible to disregard the inner workings of the layer below (Youseff et al. 2009). The ability to abstract is of importance when it comes to interoperability because it allows an organisation to focus on a well-defined part of the services that need to interoperate, while being able to safely assume that other parts "just work" according to predefined parameters. When describing a system using a layered model approach, it is important to strike a good balance between the generalisation of that which can be made abstract, while at the same time being specific enough to allowing specificity when an application of the model requires customisation to fit a certain problem or implementation.

There are no formal requirements to what constitutes a layered model, and exactly how the different layers must interoperate. For example, in the OSI model, it is possible to imagine services (e.g. Management) that can span multiple or all layers, and some models may have services that bypass one or more layers. That said, the layers and services should be well defined, and the model must be logically coherent.

7.3 Mapping OAIS Services to a Layered Service Model

The basic information types in OAIS can be arranged according to information complexity. There is an increase in complexity from the relatively simple Digital Object to the more comprehensive Information Package. This progression from simple to complex is comparable to how information flows in a layered model, where information in one layer is used, manipulated and passed to a higher layer. This model reflects the development in complexity of Information Objects corresponds to the document lifecycle, where a document goes through a number of stages over time (Jia-sun 2001). This progression has been used as the basis for the model presented in this research.

7.4 Description of Layer Functionality

The following section will explain how the layered model works from a systems and data perspective.

7.4.1 PaaS Layer

At the bottom of the model is the PaaS Layer. This layer provides storage for ARM systems or business systems and provides storage for common objects such as strings, numbers or files. The PaaS layer can be seen as a combination of the PaaS and IaaS service models from the NIST definition of cloud computing in that it includes the provision of processing, storage, networks, and other fundamental computing resources, including the ability to deploy software such as operating systems. It should be stressed that this does not include the actual cloud hardware (servers, routers and so on), but it does include the ability to exercise control over operating systems, storage, and deployed applications.

The PaaS Layer will typically utilise a shared infrastructure backend such as Amazon EC2 or other XEN based hyper-visor. The PaaS Layer provides storage for bit-strings, that in higher layers will be interpreted as Digital Objects that are the target for archiving and also for system data (Kurth 2013).

The PaaS Layer includes the ability to ensure that data is secure at the bit-level, guaranteeing that data does not suffer from media failure, update errors or "bit decay". This is done via error checks, performing backups and adding fixity checks (for example using CRC) to Digital Objects to protect these against tampering. The PaaS Layer can take advantage of the shared platform it is installed on to serve as storage for several applications. Each Digital Object is made up of one or more bit-strings, identified by a URI makes it available to systems in higher layers. In other words, a Digital Object is a primitive unit that makes no sense as information outside the context of a system that has the functionality to read and represent it.

To ensure the layer can serve as a backend for applications in the SaaS Layer, the PaaS API need to be specified as a Service Provision. There are currently a number of efforts underway to standardise cloud APIs, but most cloud platforms still make use of different APIs. One common trait is that most of these

platforms interact with application via some kind of RESTful web API, such as SOAP. No matter what API is used, it needs to be documented.

7.4.2 SaaS Layer

The systems in the SaaS Layer can either be locally hosted or SaaS offered via the cloud. This layer holds the business applications that are responsible for representing bit-strings from the PaaS Layer as documents that can be accessed and understood by a user or other systems. An example of such software is Google Docs that sit on top of a Platform and a storage back-end hosted by Google. When using such software, users are presented with documents and the tools to edit these. The software can have functionality allowing it to export and represent documents in a number of standardised formats, such as PDF, Word, RTF or Text. These object will be referred to as Content Data Objects (CDO) In order to distinguish them from the Digital Objects (DO) stored in the PaaS Layer. Content Data Objects may consist of several Digital Objects aggregated into a format that can be handled by systems in the SaaS Layer. It may be a file such as a MS Word document with accompanying metadata or a data table stored in a database. For example, in a SaaS word processing application, a document consists of text, formatting data, representation data etc.

In order to uniquely identify documents or records that may become the target for preservation, a different URI also needs to be assigned to identify individual Content Data Objects.

Once a document is declared as a record targeted for long term-preservation, it needs to be transferred to an archive system. In the presented model, this means making the Content Data Object and its metadata available for harvesting by the Preservation Layer. However, as opposed to the OAIS Model, none of the data ever leaves the PaaS Layer, but is stored by the business application as bit-strings in an agreed format in a dedicated cloud storage location. In other words, what is transferred from the SaaS Layer to the next layer is Content Data Objects and the metadata necessary for the creation of packages for preservation along with metadata providing access and pointing to Digital Objects in the PaaS Layer.

7.4.3 Preservation Layer

The Preservation Layer provides services used to ensure that Content Data Objects can be managed and interacted with by ARM systems. It also provides metadata facilitating long-term preservation. Finally, it provides functionality for handling different information types necessary for the creation of packages. In order to guarantee that this information is usable to the above layer, package information descriptions and the ways of interacting with the Preservation Layer must be fully documented.

There are four information types in this layer necessary for package creation and preservation: Representation Information, Preservation Description Information, Packaging Information and Package Description. These information types have their origin in the OAIS model, and comprise in themselves an Information Object stored as bitstrings in the PaaS Layer with a unique ID, allowing it to be referenced when creating Information Packages.

Since the objects in the Preservation Layer are metadata there must be schematic rules for their use. This thesis does not mandate any particular metadata schema for preservation to the exclusion of another. However, it is important that the metadata is understandable and usable for the systems in the Interaction Layer above.

Representation Information

The purpose of Representation Information is to ensure the understandability of Content Data Objects to a Designated Community with a designated knowledge base. Representation Information comes in two types, structure information (explaining data structure concepts such as file types, encoding etc.) and semantic information (explaining the terminology and language used). One important aspect of Representation Information is that it may need to be referenced by other Representation Information. The OAIS reference model uses the example of a series of bits representing an ASCII table, where the representation information includes a definition of ASCII.

Whereas it may not be necessary to provide Representation Information Objects for open international standards such as ASCII or ODF, a central register of Content Data Object representation information types should be kept. This can take the form of a Representation Information Registry (RIR), i.e.

"...a systematic collection of representation Information Objects or locatable references to objects held elsewhere. The RIR exposes these objects for discovery and processing by human or automated systems. RIRs may be designed to describe any class of representation information, or may specialize in a particular class, such as file formats" (Giaretta et al. 2005).

With a Representation Information Registry in a cloud based system, it becomes possible to specify representation information for used Content Data Object types and store this in the below layers. This allows systems managing preservation to simply refer to stored representation information instead of providing it each time an Information Package is passed to the archive layer. Such a system takes the burden of providing representation information away from individual Interaction Layer systems, but requires that all Content Data Object types that are potential targets of preservation are registered in the Representation Information Registry.

Preservation Description Information

Reference Information is one of four types of Preservation Description Information. It is a unique identifier that allows systems to refer to a particular Content Data Object. It can be a Record ID Serial Number or similar. In the SaaS Layer, a URI is assigned to identify all individual Content Data Objects. This URI follows Content Data Objects in the above layers, and serves as Reference Information.

Provenance information relates to the history of a Content Data Object and the changes it has gone through during the stages of the document lifecycle up till and including archival. Depending on the content to be preserved there can be a number of different kinds of Provenance information.

In order to guarantee that provenance information about Content Data Objects from systems used in the early stages of the document lifecycle is usable for preservation purposes, it is necessary to define a format for encoding the information and passing it to the above layer.

Context information describes the relationship between a Content Data Object and its environment. Like the Provenance Information above, it needs to be provided by the SaaS Layer in an agreed format.

The Preservation Layer must also add Fixity information to Content Data Objects to ensure that these are not tampered with. This is functionality is similar to the fixity checks of Digital Objects in the PaaS Layer. However, in the Preservation Layer, fixity information is added to Content Data Objects and is stored as an Information Object in the PaaS Layer (The OCLC/RLG Working Group on Preservation Metadata 2002).

Packaging Information

Central to the OAIS Model is the concept of Information Packages. Records are ingested into an OAIS as SIPs, stored as AIPs and is provided to consumers as DIPs. These basic concepts can still exist in the layered model, for example if records are transferred from one ARM management system to another and these use different information formats. Here, creating DIPs and SIPs may be necessary. What is different, however, is the contents of the packages. Functionality provided by one layer can be utilised by one or more entities in the above layer. This means that one Information Object can be used in different contexts. Since all Information Objects have URIs and can be accessed by several systems, the need to include all this information in Information Packages disappears. As an example, instead of adding one or more pieces of Preservation Description Information to an Archive Package, it would be enough to use an URI to point to where the PDI bit-string is stored.

In the OAIS model, SIP packages arrive from a Producer and are ingested into the archive. At the time of Ingest, SIP packages do not necessarily have complete Preservation Description Information and Representation information. This is not the case in the layered model, where the Preservation Layer provides this information. Content Data Objects to be archived are presented to the Interaction Layer from the Preservation Layer containing URIs pointing to complete Preservation Description Information and Representation Information.

This means that it is possible to create complete Information Packages based on the information in the Preservation Layer. Using Packaging Information provided by the Preservation Layer, systems in the above layer can manage the three package types used in an OAIS archive, namely SIPs, AIPs and DIPs.

Package Description

The last information type in the Preservation Layer is the Package Description, which provides searchable information about an Information Package and makes it available to access aids.

In the model, the different information types in the Preservation Layer are shown in a way that reflects the information flow in most organisations. There may not be any need to have preservation information at the beginning of the document lifecycle, where information is still stored in a CMS. However, there is no reason why preservation information cannot be provided as a service in earlier stages of the document lifecycle. Such a model is certainly possible, provided there is a specific business need, and if the handling of such information is supported by the systems in the Interaction Layer.

7.4.4 Interaction Layer

In this layer are the applications that provide access to archived resources, based on the needs of the organisation. The purpose of these systems is to provide a point of interaction with the different Content Data Object types defined in the SaaS Layer. The systems in the Interaction Layer can have different functions, and organisations either use existing systems or custom design systems based on compliance requirements such as those specified in MoReq or DoD 5015.2. Because of functionality provided by the underlying layers, systems only have to meet a subset of the functionality in any standard, as requirements for backup, storage, preservation etc. are already covered.

Systems in the Interaction Layer must be designed to take advantage of the information in the layer below. As an example, this means that in a cloud solution where the Preservation Layer is provided by a third party, systems in the Interaction Layer must be designed to interact with it. As mentioned earlier, package information descriptions and the ways of interacting with the Preservation Layer must be documented.

As in the SaaS Layer, the systems in the Interaction Layer can themselves be installed in the cloud or hosted in local data centres. The information systems all have Content Data Objects as the object of management, and retrieve these from the PaaS Layer, based on information in the Preservation Layer and using an agreed protocol for data transfer to access the PaaS Layer, such as SOAP. When it comes to systems, the biggest difference between current practice and

the proposed model is how they access stored data. In non-cloud systems, each system uses its own method of storage, whether this is a database or a simple file directory structure. In the model, records management systems share common functionality via layers, making a scenario with multiple ARM systems managing the same information possible.

As mentioned previously, if digital ARM services were to be offered in the same way as cloud services, organisations would be able to choose the level of archiving functionality needed for a specific purpose. As in a cloud system, services would be abstracted in such a way that the organisation in question could choose a certain level of service, trusting the cloud to provide the underlying functionality. Such a model of abstraction could be described by dividing the functionality offered into layers, with one layer dependent on the services provided by the one below.

The previous section explained the services offered by a digital archive according to the OAIS reference model, where different functional entities are responsible for providing services. However, the OAIS functional entity model does not integrate well with a layered service model. The reason for this is that the functional entities in OAIS are interdependent. As an example, in a scenario where an organisation is looking for a storage solution offering bit-level integrity for Data Objects to use as a back-end for an archiving system, this would involve overlapping functionality from Data Archiving and Archival Storage.

In the OAIS, it is the functional entities that are responsible for providing archiving functionality, and as such it would be ideal if these were directly comparable to layers in a layered model. Since this is not the case, the question arises whether it is possible to create a model that uses functionality and concepts from OAIS while still applicable in a Layered Service Model.

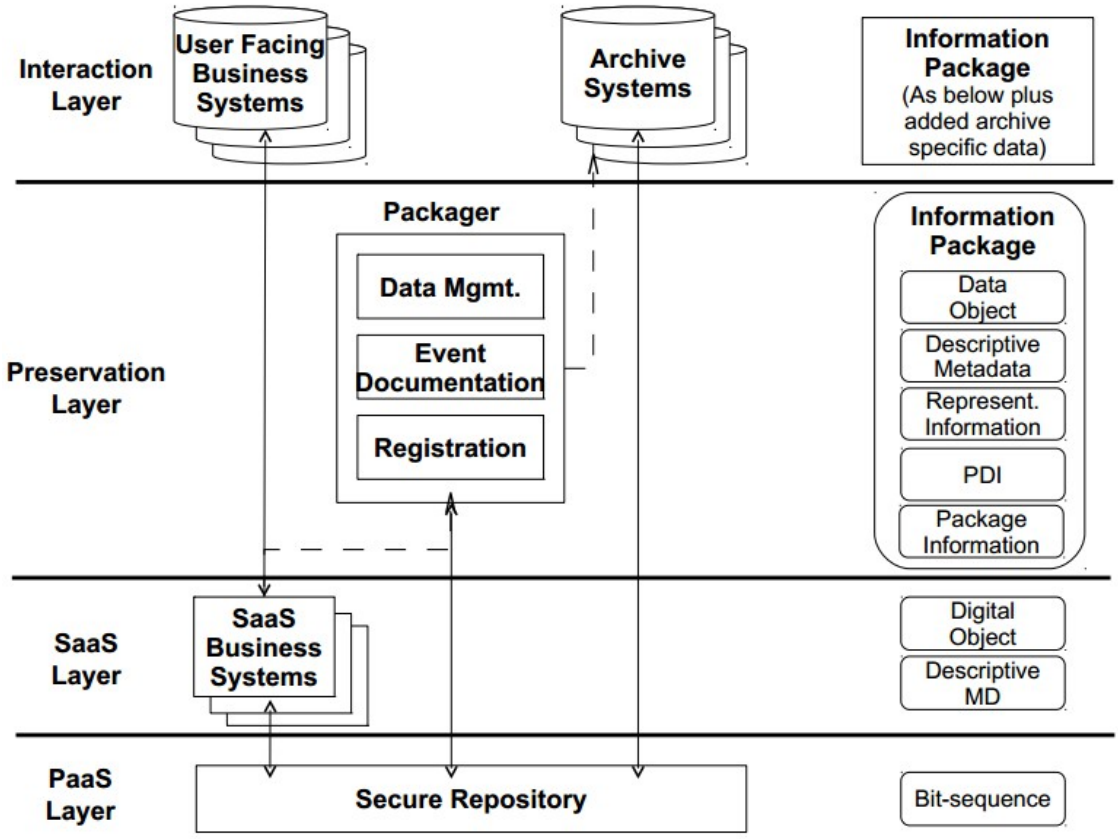

Figure 4. Cloud system and information flow

7.5 Information Flow Example

To illustrate how the model may be applied in the real world, the author has created a scenario presenting a cloud system using the model. The scenario shows how an email with an attachment created in a SaaS System passes through the Preservation Layer and is stored in an archive system.

In the presented system, a typical information flow would be as follows:

1. A user belonging to an organisation creates an email, using an online email client (SaaS Business System) accessed via a browser (User Facing Business System). The user attaches a HTML document to the email and sends it to a recipient.
2. Based on organisational policy, the email is declared a record in the business system. The bit-strings making up the email and its attachment are locked in the PaaS layer. An XML notification pointing to the bit-strings making up the email is sent to the Packager.

3. Based on the notification, the Packager retrieves the relevant data from the PaaS layer. After the data is validated, parts may need to be converted (for example, metadata may need to be cross-walked to a different schema). Based on the converted data, an XML file of PREMIS Preservation metadata is created based on Business System Metadata, Pre-registered information and Event Related Information. This information and the corresponding Data Object is saved as bits in the PaaS layer, as a virtual package. A notification pointing to the relevant bit-strings is sent to the Interaction Layer.
4. The archive application in the Interaction Layer receive Submission Information Packages from the Preservation Layer. Based on archive policies, such applications may also save additional metadata, such as Access Aid specific data.

8 Applying the Layered Model – A Theoretical Case Study

In order to illustrate more clearly how the proposed model may be applied, the next section presents a case study using the Japanese government as example. In Japan, the National Archives of Japan (NAJ) is charged with preserving government and state owned records as historic materials. It receives its mandate from the Public Archives Law of 1987, the National Archives Law of 1999, and the newly enacted Public Records Management Law (National Archives of Japan 2007).

Whereas the legal framework for the management of public records in Japan has improved considerably, there are still a number of problems with the way archives are managed in practice. In recent years, there have been a number of serious record management mishaps, such as the "Pension Scandal" where millions of records of insurance premium payments had missing or incomplete information, made worse by the use of several incompatible systems (El-Agraa 2009). It is hoped that the new Public Records Management Law will help alleviate some of the problems with public records management.

Further to this, The Japanese government is currently working on a new IT strategy. As a part of the so-called "ICT Hatoyama Plan", the Ministry of Internal Affairs and Communications has started a massive cloud computing project, named "the Kasumigaseki Cloud" to provide computer resources necessary for government departments through a shared platform. The ambitious project started in 2009 is expected to be finished in 2015 (Chan 2009). Because of the scope of this project, covering the entire records lifecycle and its cloud based nature; this would be an interesting subject for a case study applying the layered model.

8.1 Current System Setup

In Japan, government ministries and agencies need to keep records according to the Enforcement Ordinance Article of the Law Concerning Access to Information Held by Administrative Organs (Koga 2010). A number of isolated business systems are used for document creation and use. The paper or

electronic documents that need to be kept as records are registered in a common document management system used by most agencies. This keeps track of record information such as record title, creator, date registered, retention period etc. Records can be stored either locally or offsite as inactive records for the remainder of their retention period. At the start of each fiscal year, the Prime Minister receives reports from the heads of the administrative organs and in consultation with the NAJ determines which records are appropriate for transfer to the NAJ as historic records. Based on this, a Transfer Plan is developed, and detailed schedules for the transfer are arranged by ministries and agencies. Once this is complete, the electronic and paper records and the responsibility for their management is transferred to the NAJ (Okamoto 2010).

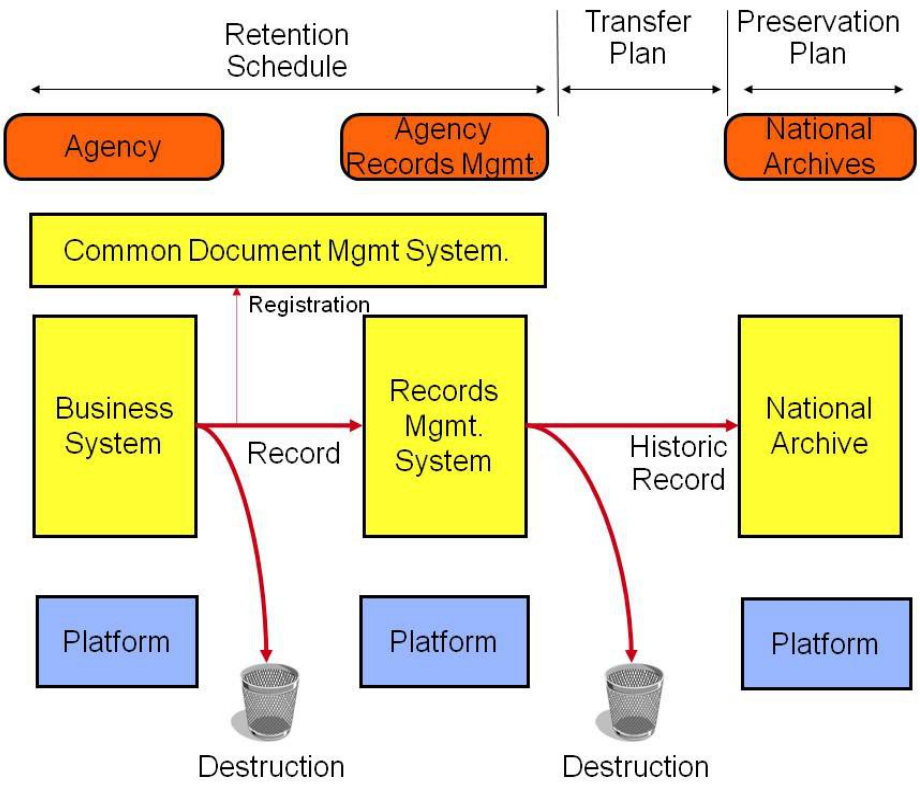

Figure 5. Overview of current processes and systems

8.2 Problems with Current System and Processes

Looking at the current procedure for archiving electronic records, there are a

number of areas that could potentially be improved by implementing cloud computing functionality using the concepts from the proposed model. There are three problem areas that require particular attention:

Lack of system integration.

At the moment, there is no direct link between the different systems in the above process. Whenever a digital record needs to be transferred from a government RMS to the NAJ, the data needs to be packaged in a format that suit the destination system. As government agencies have been free to formulate their own records management policies up till 2009, disparities between systems are to be expected. This in turn necessitates data conversion, a resource consuming task.

Lack of resources

At the moment, the burden of transferring records to the National Archives lies with agencies the records originate from. They have to prepare records for transfer according to the Transfer Plan and directions from the NAJ. Ideally, the NAJ having the knowledge and expertise could provide direct help to the ministries and agencies, however since there are only 42 people working full time in the NAJ, there are limits to the assistance they can provide. This lack of support in transferring records makes it harder than it should be for organisations to submit records for long time archiving and for the NAJ to ingest records into their collection.

Preservation

The final problem concerns the preservation of electronic records still residing in local ministry and agency systems. Because of long retention schedules, some document types have to be stored locally for many years before they can be transferred to the NAJ. This can cause problems because records can become inaccessible due to changes in the hardware and software platform in use. Of course it is the responsibility of the records management section of the originating organisation to make sure that this doesn't happen, but with the current state of records management in local ministries and agencies, it may be optimistic to expect that they have the resources and skills to do proper preservation planning.

8.3 Creating a System and Workflow Based on the Layered Model for Cloud Computing

The specifics of the cloud solution for the Japanese government have yet to be decided. However, it is part of the plan that it will not only be a hosting platform (PaaS), but also offer software services (SaaS).

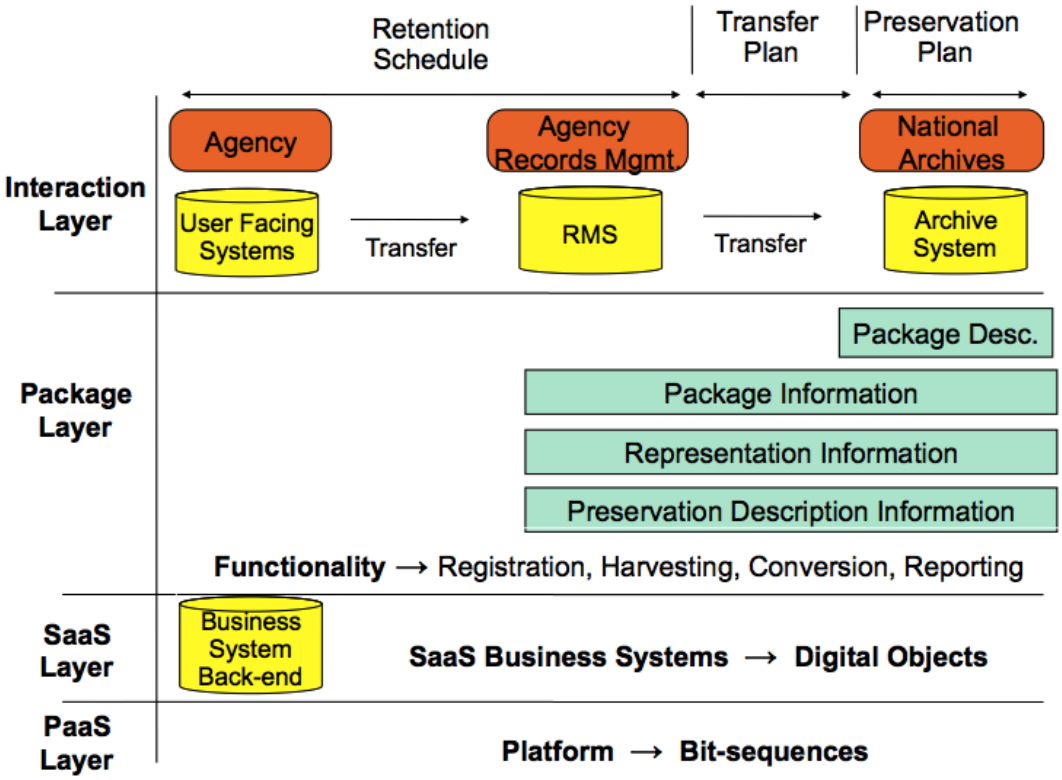

Figure 6. Cloud solution using layered model with existing process

Assuming that the Kasumigaseki Cloud is similar to existing cloud solutions, this means that it will cover the two lower layers of the model, and that an archive system would not have to take into consideration functionality in these layers. However, these layers still need to be documented and a migration path decided.

Interaction Layer

In the application of the model, it is assumed that the Japanese government will keep using its existing ARM systems, after a migration to the cloud. This means that the Interaction Layer will have 1) an interdepartmentally shared DMS for registering business documents, 2) a number of RMS for the storage of

business records until the end of their retention period and 3) an archive system for the long time storage of historic records. Since both Preservation Layer and the parts of the Interaction Layer dealing with accessing records from the PaaS Layer will have to be custom built, this provides the Japanese government with the freedom to ensure that either layer is compatible with the other.

Preservation Layer

The Preservation Layer has several functions. First of all, it is a middleware application that allows systems registered in the Interaction Layer to access the PaaS Layer. It also harvests records and metadata from the SaaS application and creates packages. Figure 6 shows that Information Packages are created at the time when records are added to the RMS. At this point records declaration has taken place, and preservation becomes important. At the time when records leave local RMS and are transferred to the NAJ, Package descriptions need to be made available to allow consumers to search and for the creation of DIPs.

1. **Registration**

The Registration entity is an application that manages the registration of individual Business Systems and RMSs used in government departments. It stores 3 types of information: 1) Systems information such as systems type, access rights, method of communicating with PaaS Layer. 2) Information about the record types created, what metadata they use and what format this is in. 3) Information used to generate preservation data. This information is derived from both the RMSs (e.g. representation information) and the SaaS Layer (information used to generate archive Information Packages.

2. **Harvesting**

Harvesting is responsible for periodically monitoring whether any new records have been added to the shared data storage and if this is the case, for presenting these and their metadata to the Conversion entity. When a record is saved, it is only a part of the saved data that is relevant for archiving purposes. Based on information stored in the Registration module, the Transfer application reads this information and forwards it to Conversion.

3. **Conversion**

The Conversion entity is an application responsible for the conversion and completion of metadata and the creation of packages that can be accessed by the Archive. When a record is saved in the RMS, the conversion entity converts the metadata into the same schema and format used in the Archive based on information stored in Registration, e.g. by using metadata cross-walking. It may be necessary to add extra metadata fields in case these are required by the Archive system. In such cases this metadata will be added as generic fields based on information in the Registration entity. Once complete, the new metadata is saved in addition to the original RMS metadata.

Once conversion is done and the record and metadata is saved in the database, a log of the process is sent to reporting. Once this is complete, the record is available to the Archive system for search and access.

4. **Reporting**

Reporting is responsible for collecting information about completed actions from other Preservation Layer entities and forwarding these to the systems in the Interaction Layer. Collected information can be logs, errors or other system notifications. The format and amount of information to be collected depends on the requirements stipulated in the Interaction Layer systems.

8.5 Evaluating Remarks

The previous section describes a case study of how the model may be applied, using records transfer from Japanese government agencies to the NAJ as an example. Not much is yet known about the proposed Kasumigaseki Cloud, and the case study presented here is no more than an example of one possible solution. With a large number of legacy systems and an entirely new cloud platform, designing a workable solution will be a big task, whatever model is used.

9 Application Profile Design for Cloud Archiving Systems

9.1 Functional requirements for an Application Profile

As stated in the introduction, the goal of this research is to build a complete archiving solution that can be deployed using cloud technology. To that end, it is vital to define guidelines the use of metadata. Whereas the layered model defines the types of metadata that belong to a layer, it is not a metadata schema, and it does not specify any rules or restrictions for metadata creation and use.

To provide guidelines for data transfer and package creation, an application profile can be be created for this purpose. The Dublin Core usage glossary defines an application profile as:

"A set of metadata elements, policies, and guidelines defined for a particular application. The elements may be from one or more element sets, thus allowing a given application to meet its functional requirements by using metadata from several element sets including locally defined sets..." (Woodley 2001)

9.2 The Singapore Framework for Dublin Core Application Profiles

As interoperability is an essential part of the proposed cloud solution (referring here not only to interoperability between producer and archive but also to potential interoperability between different digital archives), the author chose to design the application profile using the Singapore Framework for Dublin Core Application Profiles (Nilsson et al. 2009). Even though the title of the framework seems to suggest that the framework is only applicable to Dublin Core profiles, it is in practice possible to use the model for development of any application profile where it is important to define how metadata properties are defined in statements and how it should be constrained when it comes to the use of syntax and encoding.

The DCAP is modular in its structure. It consists of a number of components, split into three layers, with the lowest being the so called Foundation Standards,

RDF and RDF/S and the one above being Domain Standards, namely Community Domain Models, Metadata Vocabularies, the DCMI Abstract Model and the DCMI Syntax Guidelines. All of the previous components are models and domains already in use by various communities. The structure of the Application Profile itself will be explained in the next section (Coyle & Baker 2009).

To conform with the recommendations of the Singapore Framework, there are a number of components that must be present in an application profile. These are:

1. **Functional requirements (mandatory)**

The functional requirements define the purpose of the profile, or the problem to be solved. For the purposes of this research, the functional requirements must describe the metadata elements of an Information Package for the use of a digital cloud archive. It must provide a common vocabulary and define any restrictions on metadata terms.[3]

2. **Domain Model (mandatory)**

The Domain Model is used to show the basic entities and relationships in the domain described by the application profile. It establishes a common understanding of what the application profile covers. A conceptual 4-layered model has already been presented in section 7. This model is used as the basis of the Domain Model.

3. **Description Set Profile (DSP) (mandatory)**

A Description Set Profile is designed to create constraints on metadata. The DSP defines the metadata element sets (and metadata elements from these sets) used in the application profile.

4. **Usage guidelines (optional)**

Usage guidelines describe how the application profile should be applied. In this thesis, the usage guidelines have not been described as part of the application profile, but information about application can be found in section 6

3 The application profile presented here is later used to design an ontology for cloud archives. There is a large degree of overlap of purpose between the two. For a more in depth description of requirements, see section 10.

5. **Encoding syntax guidelines (optional)**

Syntax guidelines define how and in what format the metadata elements should be encoded.

9.2 Defining a Domain Model

As mentioned above, figure 4 presented the conceptual model that serves as the starting point of the research undertaken here. The model has been built on by adding basic functional entities, actions and relations from the functional requirements and previous research. The result is the Domain Model shown in figure 7 below.

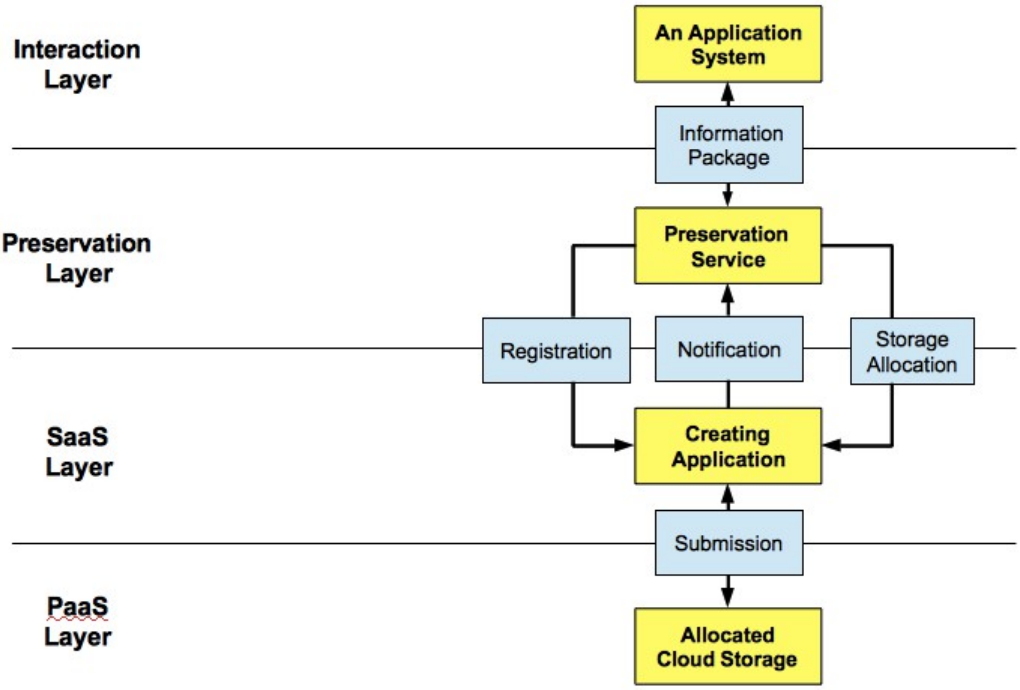

Figure 7. Domain Model specifying functional entities, actions and relations.

In the model, Business Systems register preservation related information (system information, metadata schemas etc.) with the Preservation Service. The Preservation Service sends back a confirmation with information used to access Cloud Storage. When Digital Objects are declared records in a Business System, they are sent to the storage controller and saved in Cloud Storage. At that time, the Storage Controller creates a permanent generic URI for the Digital Object and metadata in Cloud Storage. A save confirmation is sent back to the

Business System. When the save is complete, the Business System sends a notification to the Preservation Service containing the permanent URI of the stored objects, along a predefined amount of metadata for preservation purposes (specifically, metadata that has not been pre-registered and is not provided by the Preservation Service). The Preservation Service then creates a generic Information Package by carrying out the following steps:

1. Collating and normalising saved and received metadata.
2. Adding preservation information (based on registered information, event information, previously registered Business System information and external Data Sources).
3. Creating an Information Package using the resulting metadata.
4. Adding a package description.

A number of areas are out of scope for the profile. These are metadata in schemas or formats that have not been pre-registered and metadata from external schemas that do not have proper documentation. It is important to stress this, as it must be understood that whatever the finished cloud archive looks like, it can only deal with well defined metadata.

9.3 Description Set Profile

A DSP constrains the resources to be described, their properties and how values are referenced. According to the DC website, a DSP uses a number of existing metadata vocabularies and applies syntactic and formatting restraints on these.

9.4 Metadata Element Selection

To create Information Packages and transfer data, a container format for metadata transmission is needed. A suitable choice is the METS Metadata Encoding and Transmission Standard (Gartner 2002). This is a widely used and actively maintained standard, expressed in XML.

METS consists of a number of different sections, and for the purposes of this research, all but two sections have been chosen. The sections that lie outside the use case are: Structural Links (used when archiving web-pages) and Behaviour metadata (used to describe the behaviour of executable objects).

The metadata has been divided into the following sections, following the METS guidelines:

1. A Root element common for all data objects, providing a unique identifier assigned by the Storage Controller along with XML and Namespace definitions.
2. A Header element describing the background of the METS element, such as the created/modified date and name/URI information about the entities responsible, and finally the status of the package.
3. Descriptive metadata from the originating Business System, represented by Dublin Core.
4. Administrative metadata, consisting of cloud-related and other technical metadata coupled with provenance information. The Administrative metadata section is also where PREMIS metadata is included.
5. File inventory, listing the files stored in the cloud that comprise the virtual METS package.
6. Structural map, showing the hierarchical structure (if any) of the objects listed in the File inventory. All sections contain a combination of a subset of the existing METS metadata elements and elements that have been created manually.

For PREMIS, all metadata for Object, both mandatory and optional, excluding container elements were chosen. The Dublin Core Metadata Element Set is used to represent Business System metadata.

9.5 Container and Schema Selection Using METS

METS was chosen for a number of reasons: It allows the inclusion of other metadata schemas, it can express structurally complex objects and several solutions using METS already exist.

To represent metadata for preservation, the author has chosen the PREMIS metadata dictionary (Gartner 2004). PREMIS defines core preservation metadata (semantic units) needed to support long-term preservation. Whereas there are other schemas for archiving metadata, such as the UK National Archives, Requirements for Electronic Records Management Systems Metadata

Standard (UK Public Record Office 2002), PREMIS benefits from having a documented data dictionary in XML that has been made to work with METS.

Business Systems can use any number of schemas, as long as they are clearly defined and registered with the Preservation Service. The Dublin Core Metadata Element Set version 1.1 was chosen to represent Descriptive Metadata as this is an international standard and can be applied to a wide range of Digital Objects (Woodley 2001).

9.6 Defining Metadata Constraints

As the final part of the Description Set Profile (DSP), the author defined constraints for all terms in the METS profile. The following constraint types were used: Mandatory (y/n), Repeatable (y/n), Controlled Vocabulary/Free Text/Container.

9.7 Design Decisions for Implementation When Using PREMIS with METS

When designing a METS profile, especially one that incorporates external schemas, a number of decisions must be taken to ensure optimal usability and interoperability (Dappert 2008). The Library of Congress "Guidelines for using PREMIS with METS for exchange" were used as a guideline for implementation (PREMIS in METS Working Group 2008). The guidelines help in explaining a number of decisions that need to be taken by an organisation when integrating the two schemas. For example, which METS sections should be used for PREMIS metadata elements. In this case, the most challenging questions to answer have been the following:

1. **Choosing between overlapping elements from different schemas.**

A number of metadata elements in METS and PREMIS have similar functionality. For example, METS specifies *mime type* as an optional attribute of the File section, whereas PREMIS uses the more granular *format.* 8 elements were identified from both schemas that have similar purpose. These were chosen between on a case-by-case basis, trying to reach a balance between specificity and simplicity.

2. **How to deal with locally controlled vocabularies.**

It is likely that business systems may use their own metadata vocabularies. However, the only place where such metadata may be expresses is in the descriptive metadata section of a METS package. Such metadata may be an important source of information and should be preserved. For documentation and representation purposes, all external metadata schemas should be documented and registered, preferably via the use of namespaces.

3. **How to represent structural relationships between complex Data Objects.**

In the profile, the Structural map section of METS is used for this purpose. For simple (non-construct) Digital Objects, a METS package consists of 3 objects: the Digital Object itself, a metadata file and the METS package file itself. For construct Digital Objects, both the Digital Object and its individual object are treated as a complete Digital Object, and a METS package is created for each one. For example, an Email with one attachment will result in three METS packages: One for the email body (a html or similar object), one for the attachment (in this example a word file), and one for the construct object (email with attachment). The structure of construct objects are expressed as a hierarchy, where root objects are level 0, subordinate objects are level 1 and so on.

9.8 Metadata Schema Representation

Before moving on to the evaluation, it would be illustrative to present an example of metadata as presented in the profile. As mentioned previously, the initial focus in the research was on the mandatory parts of the Singapore Framework, which means that the Encoding Syntax Guidelines were not developed initially. To show the initial listing of the elements in the metadata schema, we have included an example from the schema in spreadsheet format in figure 8. The entire schema in its original format can be found in Appendix 1.

Element	METS Section	External Schema Field	TYPE	MANDATORY	REPEAT	ORIGIN	Explanation	Example 1 - PDF file	Rationale for Origin
messageDigest Algorithm	Administrative metadata 2 (amdSec) - PREMIS Metadata	1.5.2.1	Controlled Vocabulary	M	N	B(reg)	The specific algorithm used to construct the message digest for the digital object.	MD5	Part of PREMIS Data Dictionary

Figure 8. An example representation of a metadata element.

The first field from the left gives the element name, *messageDigestAlgorithm.* Next is the METS section used, here the Administrative Metadata section (amdSec) of METS, specifically the PREMIS metadata. Next is the ID of the field in an external schema, the PREMIS Id. Next is the type of field, such as controlled vocabulary, container, etc. The next two fields are the restraints Mandatory and Repeatable. Following this is the origin of the element showing where in the cloud system the element has been generated. Here the origin is B(reg) meaning pre-registered metadata from the originating business system. Following is an explanation (here taken directly from PREMIS) and an example of the element. Finally is the rationale/reason why the field is included in the schema.

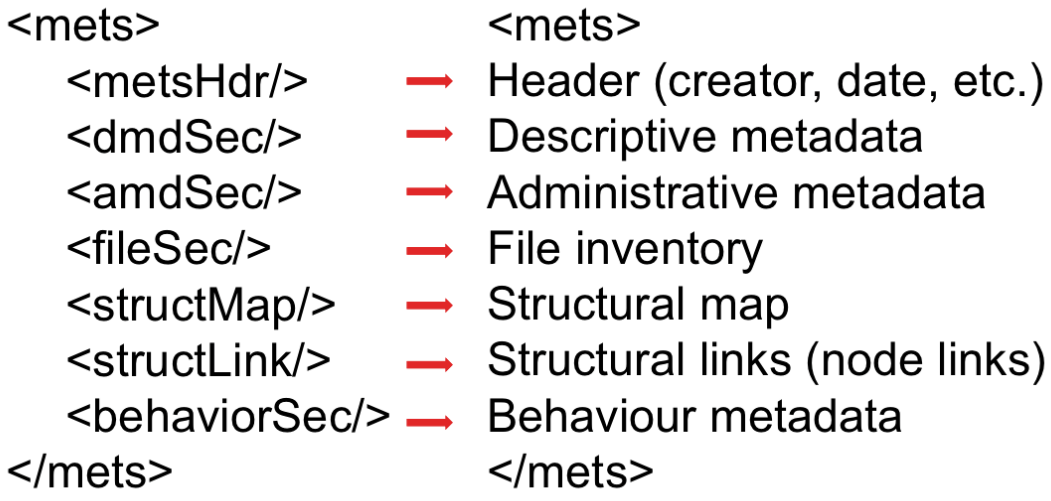

Figure 9. The structure of a METS package

9.9 Encoding and Syntax

During the design phase of the application profile, the author has been fortunate enough to find a number of related papers dealing with application profile design. Unfortunately most of these papers contain very little information about profile evaluation.

When evaluating, the most important criteria is whether the profile can solve the

problem defined in the Functional Requirements. However, whether this is the case or not can be hard to determine without first applying the application profile in the real world. By then, a lot of work may already have been put into solution, making it difficult and costly to implement changes.

The Singapore Framework lists two important success criteria: Longevity and Interoperability. Whereas these are important things to keep in mind when designing the profile (using open standards, creating good documentation etc.), both are hard to use as objective criteria before the profile has been implemented and used.

In this part of the research process, an alternative way of evaluating the proposed profile that does not require an implementation was used. One of the goals of the functional requirements is to make it easy for producers to save content and metadata to a cloud storage solution. To create an Information Package with sufficient preservation metadata, as specified in PREMIS, producers must provide a certain amount of metadata for the Preservation Service. The profile is built on a model that makes it possible to share data via a common platform and allows business systems to pre-register information. This should reduce the amount of metadata that must be provided by business systems, thereby reducing cost. To test this hypothesis, a METS package was manually created using the profile and analysed to determine the amount and sources of metadata.

9.10 Example Information Package

In the OAIS model, preparation of SIPs for submission to an archive is the responsibility of the Producer. SIPs need to contain Content Information (The Digital Object to be archived with associated representation information) and some Preservation Description Information.

Six potential sources of preservation metadata were identified: Preregistered information about the producing business system, preregistered information about the Digital Objects being submitted, the Digital Objects themselves, explicit descriptive metadata describing each Digital Object, information about events occurring during the preservation process, information from external data sources.

Using the proposed application profile, a generic METS Package was created by hand, using the following criteria: The target Digital Object is a PDF file with two related objects. The Digital Object has been encrypted and digitally signed. It has been assigned one type of Rights information. During the import process, event information from one Event is registered. The business system uses all 15 elements from the Dublin Core Metadata Element Set and for preservation all 64 mandatory and optional elements (excluding Container elements) related to the Object entity were used.

9.11 Statistics/Evaluation Based on Example Information Package

In the scenario, business systems provide 52% (55 elements out of a total of 106) of the metadata in an Information Package. The Preservation Service provides 42% (45 elements). The remaining 6% of the information is from external sources. Figure 10 shows the distribution of metadata fields according to where they are located in the METS package.

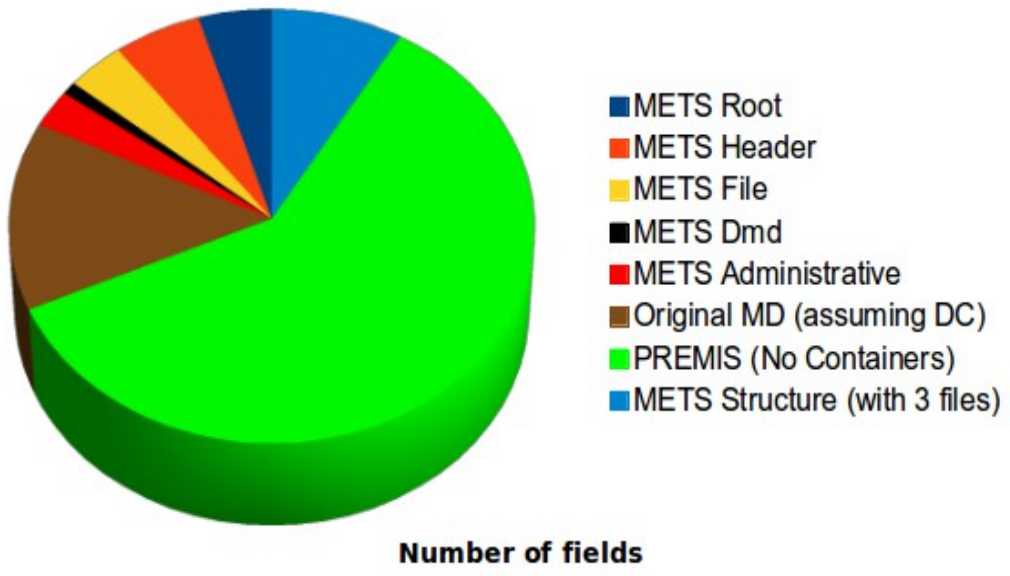

Figure 10. Information Package metadata fields belonging to different METS sections.

By examining the metadata elements one by one, it was found that 43% (46 elements) of the Information Package metadata could be pre-registered, either

by the business system or preservation service. These were mainly static elements such as those describing system information and those defining metadata types (as opposed to values). Finally, it was found that 17% (18 elements) could be auto-generated during the preservation process. An example of this is information dealing with file properties. Figure 11 shows the Information Package metadata fields, according to their origin in the cloud system.

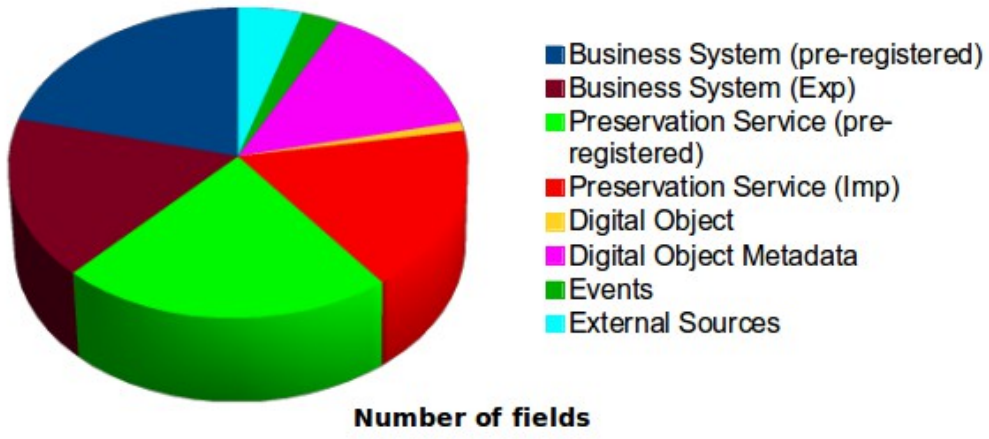

Figure 11. Metadata fields by origin

Based on these findings, it can be concluded that the application profile can simplify metadata provision for business systems, compared to systems that do not allow pre-registration. Furthermore, there is a potential for the automation of metadata provision, further reducing the amount of metadata that must be explicitly provided.

A further examination was performed on the metadata that must be provided by business systems and that cannot be automated. It was found that many of the elements described complicated attributes of Digital Objects, such as structural relations, encryption or rights information. The more complex the Digital Objects to be preserved, the more metadata must be provided by business systems, increasing cost for producers.

10 An Ontology for Preserving Digital Content in the Cloud

Developing an application profile is an important steps towards describing what system and preservation metadata is necessary in a cloud archive. However, more detailed information is needed to build an archiving system. In a cloud environment, functionality in one or more of the layers 1-3 may lie outside the control of the archiving organisation. It therefore becomes important to describe the types of data produced and received by each layer in a machine understandable format. The proposed application profile only solves part of the problem, in that it doesn't yet contain any encoding syntax guidelines, and that it is not machine readable. Without such information, it becomes impossible to abstract functionality, as there are no guarantees that the necessary data will be produced in the right format and that it will be interoperable.

10.1 Objective of Ontology

The author therefore proposes to define a domain ontology for use in the design of a cloud archive system, as outlined in the conceptual model. In order to make the ontology successful, it must do two things. It must describe the components of the cloud archive. It must describe the data objects and their related metadata at each stage of the creation and archiving process. To achieve this goal the following four main objectives were defined:

1. The ontology must provide a formal semantic model and common vocabulary using a machine-readable format such as RDF or OWL. Cloud computing makes it possible to share both data and services across different entities. Common semantics are needed to make this possible.
2. The ontology must define the model classes and their allowable domains, ranges and properties. This is necessary not only to define allowable values and qualify elements, but also to make inferences about the relationship between entities.
3. The ontology must define what data is transferred between system agents. In the presented model, Digital Objects and metadata are

aggregated from a number of different sources. Before a system can be built, it is necessary to define which agents produce which data.

4. The ontology should allow linking to metadata from other schemas. If a Creating Application produces metadata that can be reused as preservation metadata (such as the original name of a Digital Object), having a way of reusing such metadata would be beneficial.

10.2 Defining a Model Preservation System for Ontology Design

In the past sections of this thesis, a conceptual model and application profile were presented to explain how data moves between different cloud entities and layers. However, a more detailed model is needed to define the classes and properties of an ontology. To that end, a model system that exemplifies the functional entities and data types needed to create complete Submission Information Packages for ingest by an archive was defined. In reality, no two systems are alike. In the same way that the main OAIS Model does not list every possible archive entity, the number of functional entities in the model have been limited to those necessary to ensure the creation, storage, and preservation of Digital Objects. Developing a conceptual model for cloud archives, is a good starting point, but it is not detailed enough to be of much use in the actual design of a usable system. The ultimate goal of this research is the design of a cloud archive system. However, for the purposes of a simple system designed to test the ontology, the following requirements have been defined. The functionality of the system must divisible into the layers of the layered model described in section 7. It must make use of a common cloud platform, including storage and processing for Producer and Archive. The use of a common platform in this context should not only be understood as referring to the same technological platform, but to the fact that Digital Objects are securely stored in the PaaS layer, while being simultaneously available to Producer and archival applications. Finally, the system must also be able to perform automated creation and transfer of Submission Information Packages including adequate preservation metadata.

10.3 Using PREMIS for Preservation Metadata

Although there are many ways to describe Digital Objects for preservation

purposes, it was decided to continue using the PREMIS Data Dictionary for Preservation Metadata (PREMIS Editorial Committee 2011) for the ontology. PREMIS is a well established standard, maintained by the Library of Congress, with a large number of existing implementations. It is used for describing not only the Digital Objects to be archived, but also the Events, Agents, and Rights associated with them. For each entry (semantic unit) in the Data Dictionary, PREMIS defines a number of attributes, such as components, definition, constraints and applicability. It is extensive, with almost a hundred semantic units for the Object category alone.

With a standard such as PREMIS already in use, it may be tempting to believe that the goal of preservation metadata sharing and interoperability has already been achieved. This is not the case. There are big differences in existing PREMIS implementations, depending on the audience, objects to be archived, and so on (Woodyard-Robinson 2007). A number of research initiatives have been carried out to solve the problems of PREMIS interoperability, such as the PREMIS in METS Toolbox, and attempts have been made to document how PREMIS should be exchanged via METS packages (Vermaaten 2010). Nevertheless, the focus is on exchanging complete packages of preservation metadata across already established repositories. This kind of approach does not fit well in a cloud scenario, where the information being exchanged is not only packages of preservation metadata but parts of the metadata being exchanged are located in lower layers, over which the repository has no control and where changes may occur. Furthermore, PREMIS only covers preservation metadata about objects from the time of ingest into an archive, and is not related to the metadata used at earlier stages of the document life-cycle in the cloud, where any number of other schemas may be used.

To sum up, PREMIS was chosen as the basis for the ontology, again recognizing that it must be extended to cover the unique characteristics of a cloud environment and other schemas in use by creating applications.

10.4 Defining Class Aspects

As mentioned previously, the presented ontology covers both individuals (instances of classes) that relate to cloud system components and to the Digital Objects to be preserved. The classes related to preservation metadata for

Digital Objects have been taken directly from the PREMIS Editorial Committee OWL ontology draft (Peyrard 2011). These are part of the PREMIS data dictionary, and need to be included. In Figure 12, these classes have been given the prefix “pr:”. Classes that are not elements of the data dictionary have been given the prefix “cl:”. The prefixes in the figure are not meant to represent all the namespaces used in the ontology, but rather as a way of distinguishing between the classes that have been directly imported and those that have not. The entities from the PREMIS data model have been used as super-classes (Agents, Events, Objects and Rights). These entities not only provide a convenient way to group classes, they can also be used to express class inference. For example, RightsGranted is a sub-class of Rights. The classes and sub-classes in the ontology are not intended to express property inheritance. Using the PREMIS data model entities to group classes has the benefit of providing a second level of semantics, by incorporating relationship information from the PREMIS data model. For example, Agents are related to Objects, via either Events or Rights. Figure 12 shows the main classes and their subclasses. The relationships between classes, for example the relationship between an event and its outcome, are defined as properties, and are not shown in this figure. In the figure, some classes, such as Environment have a black triangle on the right side to show that there are additional subclasses.

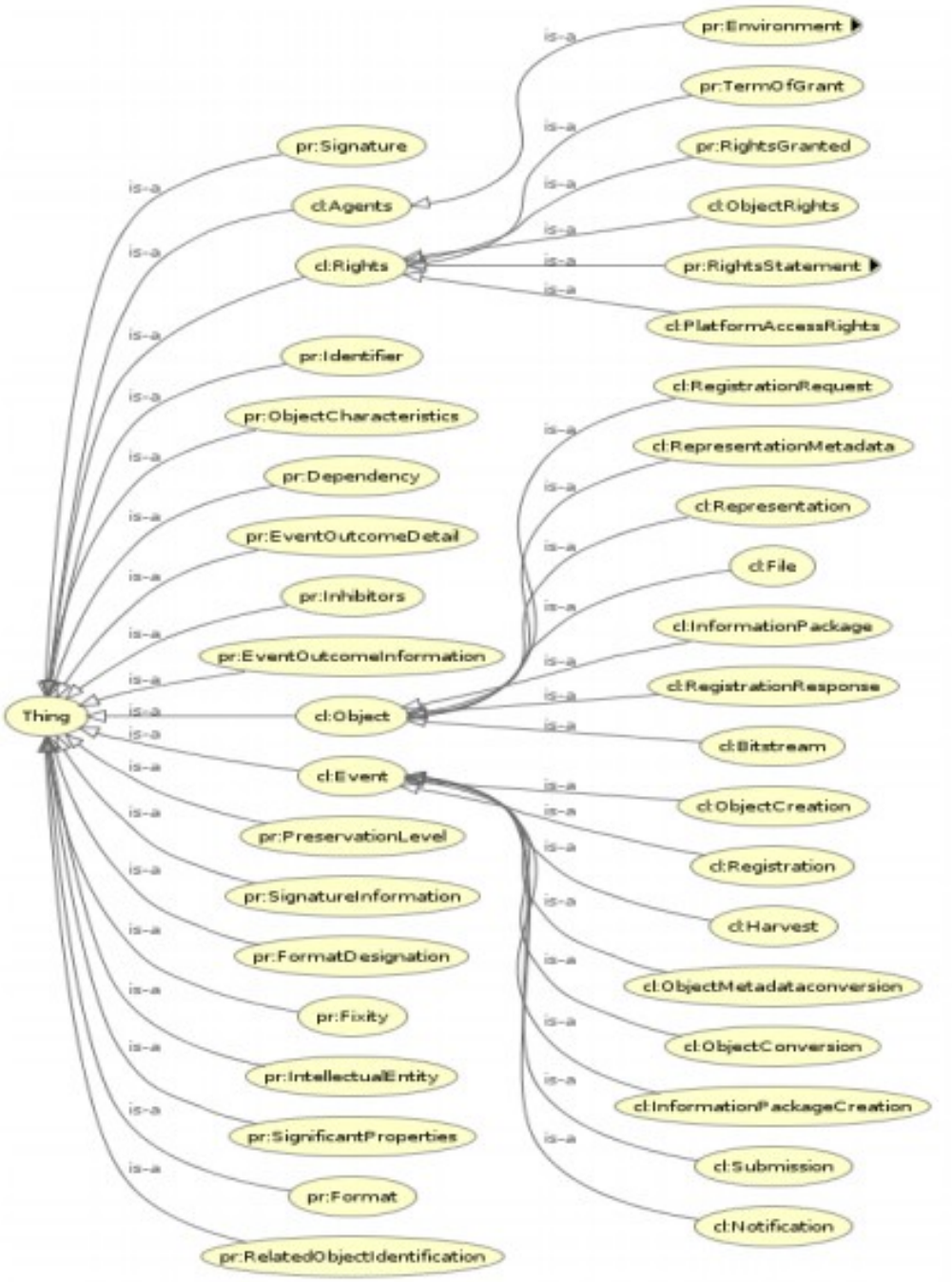

Figure 12. First 3 Levels of the Class Tree

10.5 Class Extensions and Annotations

The PREMIS Editorial Committee ontology has been extended by a number of other metadata schemas, namely FOAF, SKOS Core, PRONOM, ORE and Dublin Core. It is perfectly acceptable and even desirable to extend an ontology with extra metadata schemas; however the author has decided to initially omit

the use these for the purposes of developing the ontology. This is partly because There are no current plans to use these schemas and partly to keep the ontology as simple as possible. Another area where the ontology differs from the PREMIS Editorial Committee ontology is preservation metadata related to hardware. In the layered model, hardware (infrastructure) would be below layer 1, and as such fall outside the scope of the ontology. Some archives may wish to capture certain API information related to the producing Platform. The class Platform has been included to cover this. The terms in the PREMIS data dictionary have annotations relating to their usage, such as Definition, Rationale, Creation/Maintenance Notes and Usage Notes. These annotations have been implemented in the PREMIS Editorial Committee ontology as comments. The annotation “Layer” has been added to the classes. Layer is used to define where in the Layered Model a class is located. This is important because it makes it possible to assign responsibility for the functionality in a class to an entity in a certain Layer. For example, the Preservation Service in layer 3 can take it for granted that information relating to Inhibitors will be provided by the SaaS Layer (layer 2). One Class can be assigned to several Layers.

10.6 Object and Data Property Aspects

Whereas classes are used to capture information about individuals and groups of individuals, object Properties are used to connect individuals, and Data Properties are used to connect literals and individuals (W3C 2009). This makes it possible to express the information flow in the conceptual model. In other words, the ontology not only describes metadata values, but also dependencies, inputs and outcomes. For Object Properties (properties where the value is an individual) the following annotations have been included: 1. definition, 2. the property domains and ranges and domain/range relationship (functional or inverse functional) 3. whether the property is mandatory or not, 4. whether the property is repeatable or not, and 5. other comments such as See Also and Usage Notes. For Data Properties (properties where the value is a literal), the same information as above has been included. However, as Data Properties are used for literals, we have included an annotation for Origin. Origin is used to define which entity in the Layered Model generates the Data

Property literal. For example, contentLocationType is generated by the Preservation Service. This information is useful to determine responsibility for erroneous data.

10.7 Using OWL as a Domain Description Language.

OWL (Web Ontology Language) has been chosen to describe the domain. Compared to RDF, OWL offers better semantic expression and greater machine interpretability than RDF, and is therefore ideally suited to the purpose of creating an actual cloud system (McGuinness & Van Harmelen 2004). Furthermore, an OWL ontology for the PREMIS Data Dictionary was announced on October 18, 2011. This ontology is not finalised at the time of writing, but the groundwork in defining the PREMIS semantics in OWL has been completed. The newly drafted standard is available for comment from the PREMIS Editorial Committee, and forms the basis of the ontology (Premis Editorial Committee 2011). But most importantly, OWL is designed to work across domains, and the hypothesis is that using OWL will allow the ontology to work across the different cloud layers.

10.8 Extensibility

One of the main reasons for designing an ontology in OWL is cross domain interoperability. By having a well-defined common vocabulary, individuals from different domains can be linked according to their semantics. OWL already has three constructs to do this: owl:sameAs, owl:differentFrom and owl:AllDifferent. The author has come to the conclusion that these constructs are not enough to express the relationship between individuals in different PREMIS implementations. Good examples of this are PREMIS entities that are defined by locally controlled vocabularies. The entity may be the same, but due to differences in vocabulary use, using owl:sameAs may give rise to problems when exchanging data. The Simple Knowledge Organisation System (SKOS) mapping properties have been chosen to link individuals (Miles et al. 2005).

In the cloud ontology, there are two areas where describing the semantic relationship between terms is especially important. One is the relation between the metadata schema used in the creating application and the preservation metadata used in the Preservation Service. The other is for creating Submission

Information Packages for ingest into an archival application.

11 Putting it all Together - A Framework for a Cloud Archiving System

11.1 Evaluation of the Ontology Using a Case Scenario

It is difficult to evaluate an ontology on paper alone. The author believes that the proposed ontology fits the Domain Model, but the question of implementation and applicability to a real world archive system entities still remains. A case scenario has been designed, using existing cloud components, Digital Objects and metadata to show how the ontology can be implemented. In designing the case system, I believe that the main outstanding questions are: How can the ontology be used in the ingest and management of objects from multiple creating applications? How can data integrity and validation be insured? And how can metadata from different schemas be linked? The case scenario is very similar to the model preservation system from Figure 7. It contains the same main entities and information flow. Each entity is an individual from a class in the ontology, with the functionality of the individual explained in the class definition. Individuals are linked to one or more layers, using the Layer annotation. The individuals themselves are linked via properties, and their data properties are expressed as strings.

11.2 Registration Process

Based on the class description from the ontology, Preservation Service is responsible for ensuring the validity and completeness of preservation metadata to create Information Packages. As OWL does not specify any syntactic constraints, the preservation service provides an XML Schema registration template, to be populated by the owning organisation of the Creating Application (Joomla). Here, the class RegistrationResponse is used to define what data properties are related to the registration, and how the registration is related to other classes, such as Event outcome. Once complete, the registered data can be automatically extracted using XPath and imported into the Preservation Service. Any errors or omissions in the XML Schema will result in a negative registration response. Figure 13 shows the registration process using the

Domain Model (showing only those entities involved in the registration process).

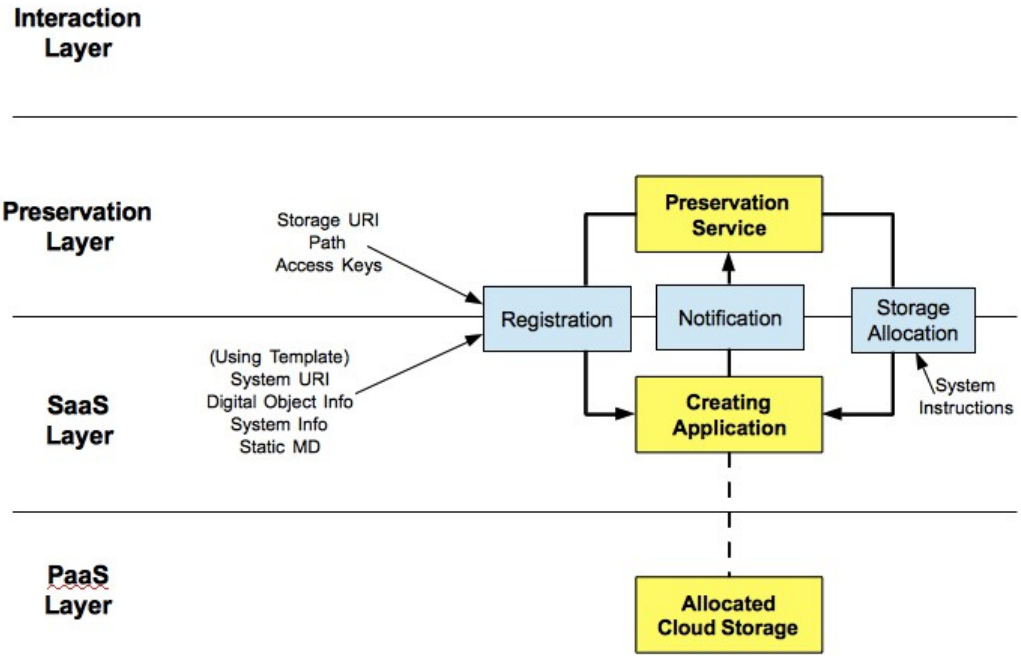

Figure 13. Registration Process part of Domain Model

The registered data gives the preservation service the ability to validate the metadata provided by the Creating Application. This is done by ensuring that all Mandatory Data Properties with the origin Business System are either preregistered (static information such as signatureMethod) or designated as provided at time of creation (dynamic information such as originalName).

If the provided data meets the requirements, a positive XML response is sent back to the Creating Application from the Preservation Service, containing access information for the shared Cloud Storage (Amazon S3), i.e. URI, path and access keys. Figure 13 shows an example of how the ontology can be used to describe the registration request being sent to the Preservation Service. The Joomla CMS is an individual from the class CreatingApplication. The functions of the CMS are described in the class annotations, which also link the class to the SaaS layer. This places the responsibility of the CMS with the SaaS service provider. The Registration Request sent to the Preservation Service is an object property, linking the two instances together. The Registration Request function is described in the Annotation property (under Comment). The Annotation property also defines the Layer in which the Registration Request is generated

(in this case, the SaaS layer). The Joomla CMS is described via a number of Data Properties associated with the CreatingApplication class, for example IdentifierURI and CreatingApplicationMetadataSchema. The Data Property values for the Joomla CMS are expressed as strings. Each Data Properly has associated annotations. These are used to describe usage, any restrictions on the values (such as whether a value is mandatory or repeatable), origin, and mapping to related classes. This information is useful for the Preservation Service for data validation. For example, if a mandatory Data Property with the Origin "Creating Application (Registration)" is not present in the registration request, the registration should not complete. Figure 5 only shows a part of the registration process, focussing on the Creating Application and Preservation Service classes using the classes and properties used in the ontology. Another way of presenting the registration request would be as an Event (rather than as a RegistrationRequest), in which case the entities in the example would be different.

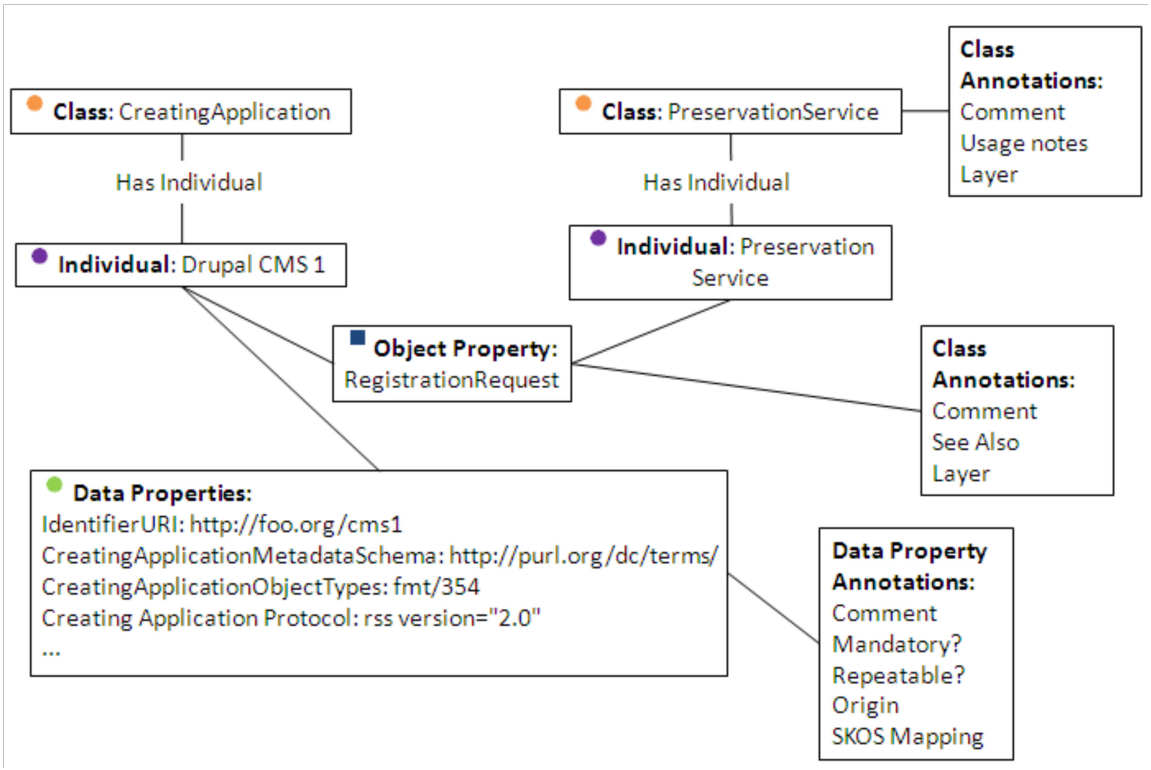

Figure 14. Part of the Registration Request Process shown using entities from the ontology

11.3 Creation of Representation and Conversion Into Generic Information Package.

Once registration is complete, the Creating Application can save digital contents to the dedicated Cloud Storage. Digital contents consist of three parts: the Digital Objects in an agreed format; original metadata such as Dublin Core or MODS, and any preservation metadata not provided during registration (dynamic metadata). Once saved, the Preservation Service provides read access to these objects for the Creating Application and the archival application (DSpace).

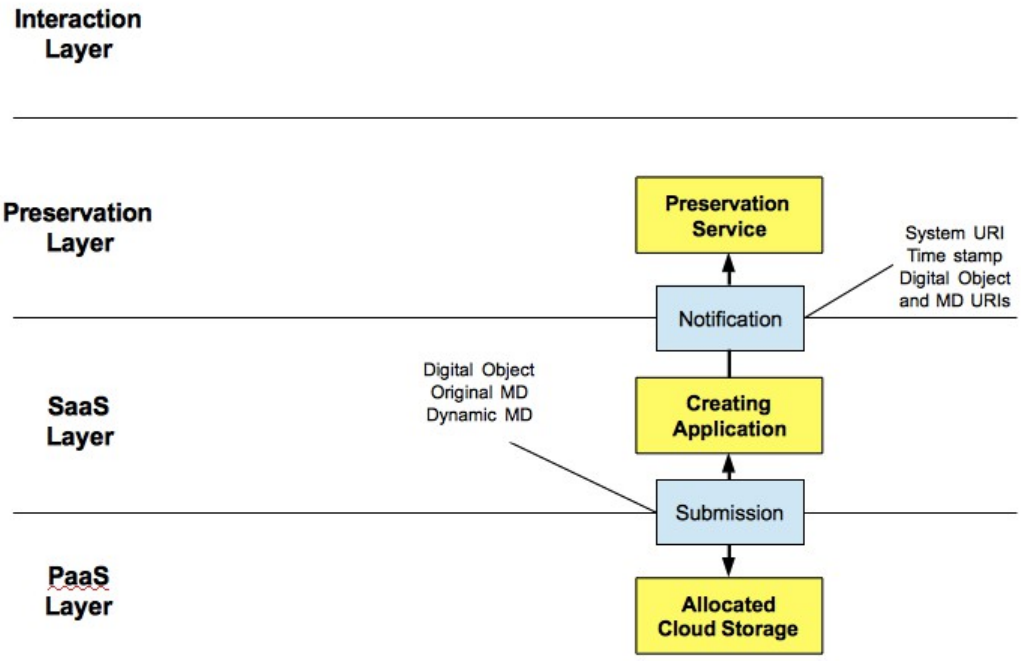

Figure 15. Save/Submission of Digital Object part of the Domain Model

Since the storage platform is shared by Creating Application and Preservations Service, the Digital Objects themselves are not included in the SIP, but only referenced as links to Digital Objects in the Amazon S3 Storage. The ontology is used to validate the metadata provided by the Creating Application at the time of save. This validation has two steps. The first is validation of quantity. If a property in the ontology with the origin Creating Application defined as Mandatory is not present after ingestion, the ingest will fail. Other errors may occur during ingest, but because each ontology literal is assigned an Origin, it is possible to determine where the error occurs. The other step in validation is

data quality. The PREMIS Data Dictionary recommends that a number of semantic units are taken from a controlled vocabulary to facilitate automatic processing. The ontology manages a controlled vocabulary by storing entries as owl:NamedIndividual linked by owl:sameAs. If a property defined as originating from a controlled vocabulary has an unknown value, the ingest fails. Another benefit in the ontology lies in the linking of metadata from different creating applications to one authoritative schema. As long as the creating applications are registered with the correct metadata linking to the data properties in the ontology, complete Submission Information Packages can be created from applications with different metadata schema. Finally, the ontology can also be used to link original metadata from creating applications to PREMIS metadata. This is done using the SKOS properties skos:closeMatch, skos:exactMatch, skos:broadMatch, skos:narrowMatch and skos:relatedMatch. As an example, I have mapped a number of DC terms to PREMIS, using the The Digital Underground Metadata Crosswalk DUMC (Gorgan & Rushay 2009). The two mapped schemas are very different in scope, and do not map well, but they are useful as an example. In DUMC, “dc.rights” is matched with “PREMIS.rightsStatement.rightsStatementIdentfier”. This will not be an exact match in all cases, so in the ontology, skos:relatedMatch has been used to show the mapping is associative rather than exact. Figure 16 shows an overview of the model registration and package creation process.

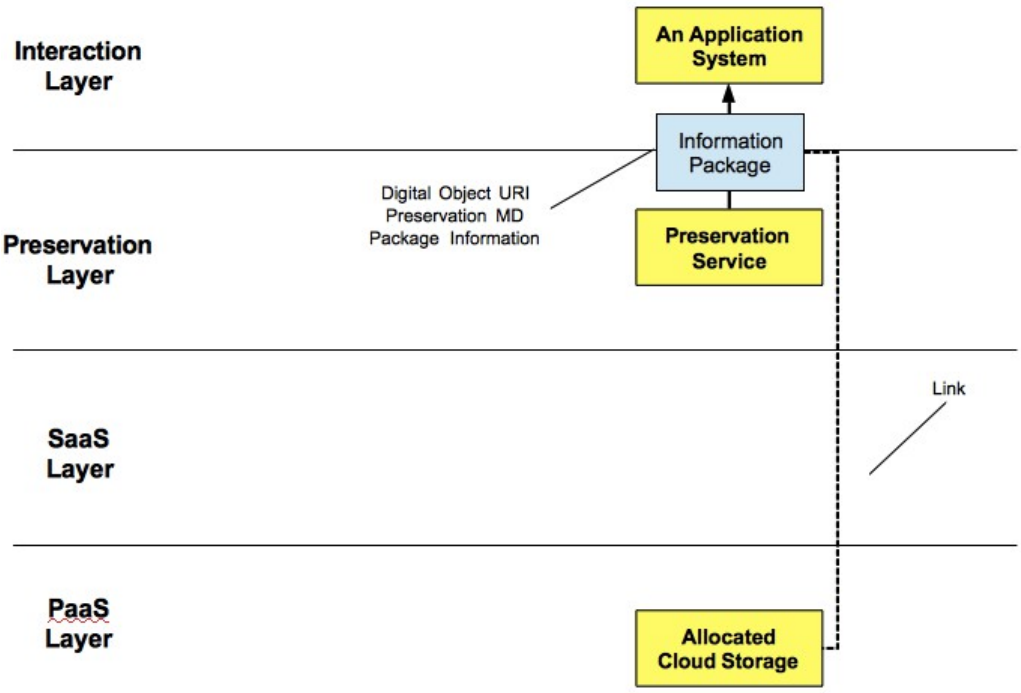

Figure 16. Package creation part of the Domain Model

11.4 Ontology Use in Validation

The previous section showed how the ontology could be used to describe both preservation metadata for cloud Information Packages, and the entities in a digital cloud archive itself. It is evident that a system based on the ontology contains a high degree of complexity. There are a number of factors contributing to this:

- **Multiple steps**

The preservation metadata needs to go through a number of steps such as submission, aggregation and potential cross-walking before it is included in a Submission Package.

- **Multiple Data Sources**

The preservation metadata originates from more than one source, as described in the Domain Model.

- **An extensive metadata set**

The sheer amount of metadata elements increases the potential for error (The Application Profile identified 106 elements).

- **Strict requirements for metadata quality**

Finally, there are strict requirements for archival metadata quality. It is difficult to correct errors in archived data both for reasons of practicality, but also because the original creator may no longer exist.

All of this complexity inevitably leads to a high margin for error in system and archival metadata. Whereas it is difficult to guarantee 100% error free Information Packages, it is possible to perform different kinds of validation before the packages are submitted to the digital archive. Because the presented OWL ontology contains rules for metadata in a machine readable format, it can be used as a powerful automatic validation tool.

To give an example of how metadata validation may be performed, testing was done on a number of manually created metadata elements from the different sources identified in the Domain Model. Figure 17 shows the validation process in its entirety.

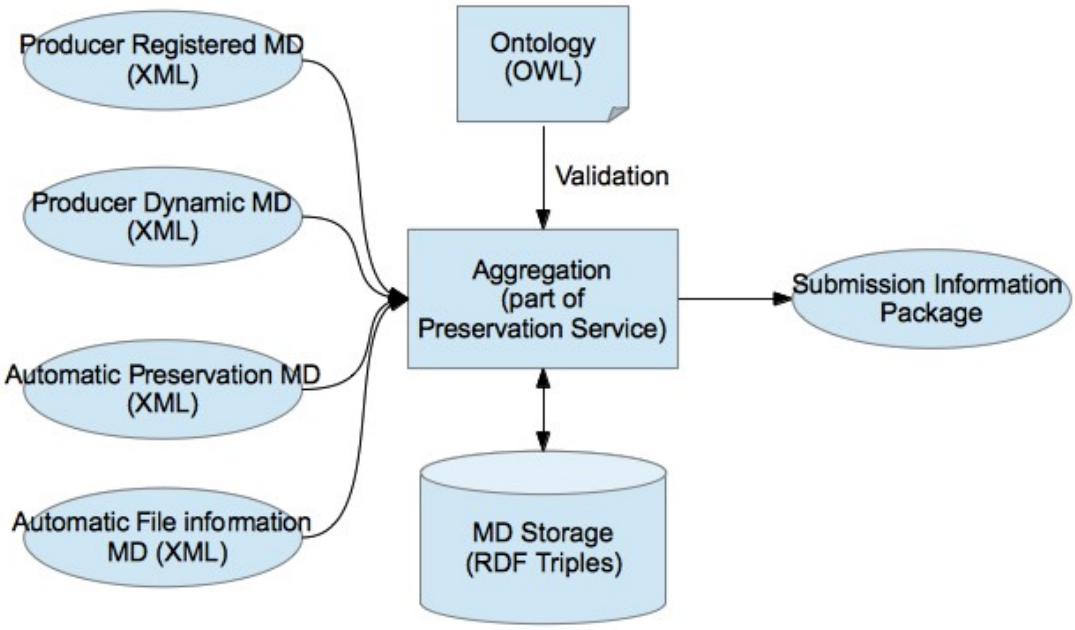

Figure 17. Metadata validation process

For each Digital Object, there are 4 different sources of preservation metadata in XML format[4]:

4 It would be preferable for the preservation metadata to be available in RDF from the start. Unfortunately, most producing applications (such as the CMS Joomla used here) only

1. **Producer Registered MD**

Preregistered by the producer using registration template.

2. **Producer Dynamic MD**

Metadata provided by the producer at time of export.

3. **Automatic Preservation MD**

Metadata created by the Preservation Service at the time of import or at other Events.

4. **Automatic File Information**

System metadata dealing with file properties, such as size, date created and extension.

The above metadata are aggregated by the Preservation Service and saved in the persistent cloud storage as RDF triples. This can be done by using XSLT on each XML file and converting the result into single RDF document, resulting in the RDF document below.

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:CloudOntology="http://www.test.org/CloudArchiveOntology.owl#">
       <rdf:Description rdf:about="http://test.org/CMS1/123456789.pdf">
       <CloudOntology:CMSURI>http://test.org/CMS1/</CloudOntology:CMSURI>
       <CloudOntology:cmsystemname>CMS_1</CloudOntology:cmsystemname>

<CloudOntology:storage_uri>CMS1.s3.amazonaws.com</CloudOntology:storage_uri>
       <CloudOntology:storage_path>/exports</CloudOntology:storage_path>
       <CloudOntology:authorization>AWS    AKIAIOSFODNN7xXj[REST    OF    VALUE
OMITTED]</CloudOntology:authorization>
       <CloudOntology:creatingApplicationOwner>Victoria
University</CloudOntology:creatingApplicationOwner>
<CloudOntology:creatingApplicationObjectTypes>fmt/14</CloudOntology:creatingA
pplicationObjectTypes>
  <CloudOntology:creatingApplicationMetadataSchema>http://purl.org/dc/elements
/1.1/</CloudOntology:creatingApplicationMetadataSchema>

<CloudOntology:signatureEncoding>Base64</CloudOntology:signatureEncoding>
```

exports metadata in XML out of the box. For this reason XML has been used in the example.

```
[REST OF CODE OMITTED]
```

Figure 18. Aggregated metadata in RDF

We validated this RDF document by uploading it to the open source Virtuoso software created by OpenLink. OpenLink Virtuoso is a cross platform database engine hybrid. It implements Web, File and Database server functionality and is able to parse and store RDF triples that can be queried by SPARQL.

The test system used the following components: Openlink Virtuoso Universal Server version 06.01.3126 running on Ubuntu 10.04 (lucid) with Linux Kernel 2.6.32-42-server running on Amazon EC2. RDF was stored using the ODS Briefcase Platform that uses a web based interface to interact with the underlying Virtuoso fileserver and triple store.

The custom RDF file shown in Figure 18 consisting of example metadata from the different sources was uploaded to Virtuoso using the ODS web based interface. The RDF XML file is automatically converted into the Turtle (Terse RDF Triple Language) code below, which is stored in the triple store (Beckett & Berners-Lee 2008).

```
@prefix ns0: <http://www.test/CloudArchiveOntology.owl#> .
<http://test.org/CMS1/123456789.pdf>    ns0:CMSURI
       "http://test.org/CMS1/" ;
       ns0:cmsystemname    "CMS_1" ;
       ns0:storage_uri     "CMS1.s3.amazonaws.com" ;
       ns0:storage_path    "/exports" ;
       ns0:authorization   "AWS AKIAIOSFODNN7xXj..." ;
       ns0:creatingApplicationOwner     "Victoria University" ;
       ns0:creatingApplicationObjectTypes      "fmt/14" ;
       ns0:creatingApplicationMetadataSchema
       "http://purl.org/dc/elements/1.1/" ;
       ns0:signatureEncoding      "Base64" ;
[REST OF CODE OMITTED]
```

Figure 19. RDF stored as Triples

Once this is complete, the RDF triples can be queried through SPARQL, either by the Virtuoso SQL API or through the ODS interface as shown in figure 20.

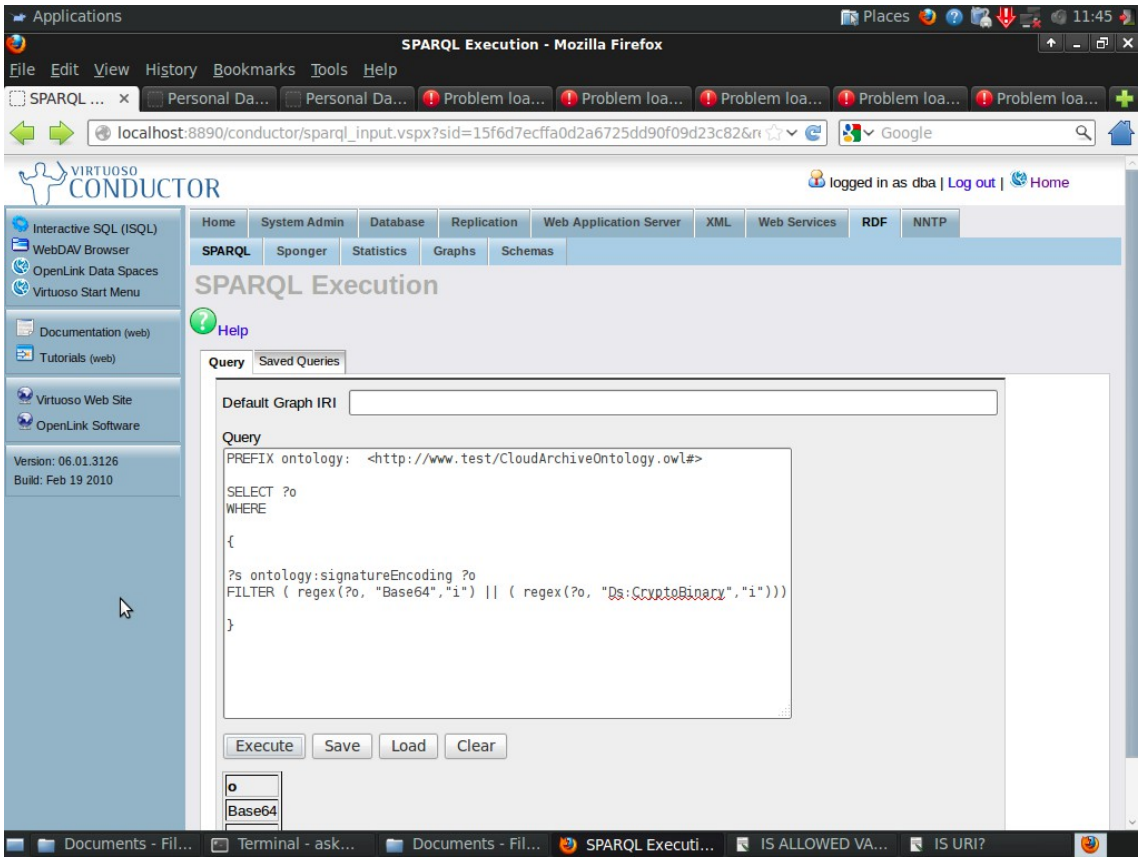

Figure 20. Virtuoso SPARQL interface

SPARQL is a powerful tool for validation of preservation metadata because it enables native querying of RDF data, coupled with the ability to query information held in different repositories (Clark et al. 2008).

With relatively simple syntax, it was possible to perform validation of the example dataset uploaded to Virtuoso. The purpose of validating the metadata is to see if there are any discrepancies between the aggregated preservation metadata and the semantic and syntactic rules established in the QWL ontology. This validation, carried out by the Preservation Service, ensures that the metadata used in Submission Information Packages is of good quality. In other words, it checks whether the Producers are submitting correct data and protects the Archival application from ingesting incorrect or malformed Submission Packages.

To test the practicality of validation, three different types of queries were performed on the stored RDF metadata.

1. **Cardinality check**

A cardinality test was performed as an example of how to discover the absence of any values defined as mandatory in the ontology. For example, to test for the mandatory term *creatingApplicationMetadataScheme*, the following query can be used:

```
PREFIX ontology:  <http://www.test/CloudArchiveOntology.owl#>
SELECT ?o
WHERE
{
?s  ontology:creatingApplicationMetadataSchema ?o
FILTER (!(isBLANK(?o)))
}
```

2. **Value Check**

Using the query below, it is possible to test whether a metadata element is using one of the allowed values from a controlled vocabulary. In this case, the element *signatureEncoding* should have the value either *Base64* or *Ds:CryptoBinary*.

```
PREFIX ontology:  <http://www.test/CloudArchiveOntology.owl#>
SELECT ?o
WHERE
{
?s ontology:signatureEncoding ?o
FILTER ( regex(?o, "Base64","i") || ( regex(?o, "Ds:CryptoBinary","i")))
}
```

3. **Regular Expression Check**

It is also possible to incorporate regular expression checks in SPARQL queries in case values need to contain a specific string. In the example query below, a check is performed to check whether the value contains *“http://”*, as a means to check whether the value is a URI or not.

```
PREFIX ontology:  <http://www.test/CloudArchiveOntology.owl#>
SELECT ?o
WHERE
{
```

```
?s  ontology:creatingApplicationMetadataSchema ?o
FILTER regex(str(?o), "^http://")
}
```

As can be seen from the above examples, it is possible to use SPARQL to validate Information Packages in a variety of different ways and to stop incorrect metadata being ingested into the archive. What is more, the ontology can also be used in locating where the error occurs. Using the Origin annotation for Datatype Properties, it is possible to see which entity is responsible for producing the error and to start planning a fix.

12 Ontology Implementation

12.1 Implementation of Ontology for the Purposes of this Research

The major criterion for evaluation has been whether the ontology can be applied to real world data. I have evaluated the ontology by using values from existing cloud system components and data from a PREMIS version 2.1 Sample Record from LoC to create instances. The cloud components used were Amazon S3 for Cloud Storage, two instances of Amazon EC2 with Ubuntu Linux 11.04 as SaaS Platform and Preservation Service platform and Joomla as Creating Application. Amazon was chosen because of the popularity of the platform, with many existing implementations and third-party plugins. Another benefit of Amazon is that they deliver both PaaS and SaaS offerings, with similar interfaces. The choice of Joomla is mainly due to familiarity from previous research. Figure 21 shows the Management Console for the EC2 instance used.

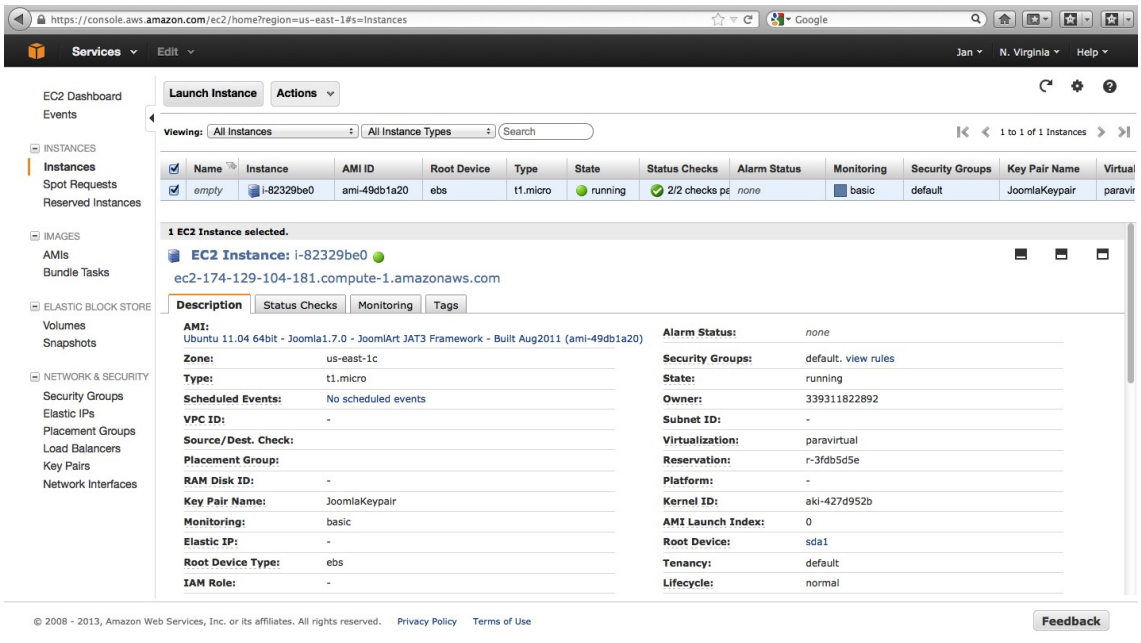

Figure 21. Amazon EC2 Management Console

The author found that the ontology was descriptive enough to create a generic XML package with PREMIS metadata, including cloud specific entities such as platform descriptions. It was possible to map instances to cloud layers, and to assign them to ontology classes. Using the OWL Object Properties it was also possible to show relationships between entities, such as which Agent is

responsible for the creation/transfer of which Object. Finally, the SKOS properties allowed us to link a number of elements from the DC Metadata Element Set to PREMIS. Another finding of the evaluation was that the main strengths of the ontology is in the description of a system for creating generic packages with preservation metadata, rather than in describing complete archive systems cover all the functions described in the OAIS reference model. The author has concentrated his efforts on creating a cloud solution that bridges the gap between several Producers and Archives by generating Submission Information Packages with complete PREMIS metadata via an automated process. There is nothing to prevent an archive from using these packages as-is. However, according to OAIS, an Archive must preserve information for access and use by a Designated Community, which in turn requires knowledge about that community. Such information is likely to differ from archive to archive, and would be difficult to automate. Based on the discussion above, I believe that the test system that was built using the ontology meets the requirements presented in the introduction of this section; it is possible for a Producer and an Archive to share a common platform, and to use data about this platform coupled with preregistered metadata to automatically create SIPs in a way that eases the burden of metadata provision for Producers. By assigning origin and layer information to each term in the ontology, it is possible to assign responsibility for the metadata to specific layers and entities. Whereas the ontology is complete in its current version (subject to modification after further tests), the system I have built for testing purposes is still not mature and relies on a number of functions being carried out by hand.

The areas that would need to be fully developed in order to create a functioning system are:

1. The registration process, where a script to automate the validation of completed XSLT template should be written
2. The automatic addition of missing metadata to harvested and validated Digital Object Metadata.
3. The automated conversion into an ingestible SIP format.

12.2 Other Types of Implementation and Relative Cost

The above section has been limited to a presentation of what system components were implemented for the purposes of research. However, the proposed model and ontology could be implemented for a number of other purposes in organisations wanting to create a cloud archive.

As mentioned in the introduction to this paper, there are a number of benefits with using cloud computing. Most digital archives are currently running on systems in dedicated data-centers, which can be an expensive investment. It is easy to imagine that smaller organisations with a limited budget wanting to host their archive in the cloud. The same can be said for organisations in areas without the necessary infrastructure to manage a digital archive independently (for example in less developed countries).

With the proposed layered model, it is possible to define service levels for each layer, according to the needs of the organisation. In other words, an organisation can decide which type of services in the layered model to invest in, according to its needs. For example, an organisation dealing with very complex objects that require special metadata, but that has no special requirements for storage, may choose to invest heavily in the preservation service, while using a basic platform for storage and computing.

By examining the data from the application profile design, and from the related research, it is possible to make some general comments regarding implementation cost. Firstly, archiving related services such as preservation and archiving applications have not yet become a commodity in the same way as PaaS, and are therefore significantly more expensive. Secondly, when it comes to the producing application, a large amount of the necessary preservation metadata can be pre-registered. However, any dynamic metadata (metadata that changes from digital object to digital object) will need to be supplied at the time the digital object is stored. If the producing application is not designed in such a way that this information can be easily obtained, it must be supplied either manually or ad-hoc, and cost will increase. Any implementation of a business system should take this into account. Thirdly, the more complex the Digital Object, the more preservation metadata is necessary. This also has an impact on cost. Fourthly, for a Producer the most laborious task in using the

system presented here is the registration process. Providing detailed information for preservation purposes will probably be difficult for some Producers, and it can be expected that they will need assistance with this task. That said, the registration only needs to be performed once, so while there may be an initial cost, it only needs to be paid once.

To conclude this section, the abstracted nature of the Layered Model brings a number of choices, when it comes to implementation, but any archive must decide what are its main priorities if it wants to keep down the cost of implementation.

13 Discussion

The research presented in this thesis attempts to form a bridge between two different ways of thinking about the storage of digital contents. One is the traditional model adopted by digital archives with the different OAIS archival entities responsible for well defined roles, such as preservation planning, ingest, data management and so on. The other is a model for cloud computing that divides services into layers and that, based on what layer the service is provided in, assigns varying degrees of control over the service to the provider (for example, with a SaaS solution, the provider has a high level of control. With IaaS, much less so).

Both these models have their merits and their place in the world. The purpose of putting them together is to make it easier for archives to build a digital archive either partly or entirely in the cloud and to know exactly what functionality and what services are available in each layer.

This merger of two models is something that the author feels is very necessary. Cloud computing is being implemented increasingly in private enterprises, academic institutions, and governments across the world. All these are potential producers of archive-worthy contents that will need to be preserved for future generations. Some governments, such as the UK and Australia are putting in place “Cloud First” procurement policies, which stipulate that agencies need to consider cloud computing as the preferred option, and only if such a solution is impossible can they proceed with the procurement of traditional, non-cloud IT services. It is unrealistic not to expect a similar development for digital archives. When such a time comes, a framework that can help create informed decisions around what functionality can be put in the cloud and what to retain in-house should prove an advantage.

Examining the model and subsequent work presented here, it is possible that a reader may question the need for such elaborate measures. After all, NIST and other organisation use only 3 service levels: IaaS, PaaS and SaaS. When looking at the top 3 layers of the model presented in this thesis, they may be easy to dismiss as “Being all SaaS”. This may be true to some extent, but in the opinion of the author, it is not a helpful comment. Why? Because it does not

solve the underlying problem of how an Archive and a Producer located in the SaaS Layer interact with the PaaS Layer. The proposed model builds on the concept of service levels, but extends this by adding layers that adopt concepts from the OAIS model, where there is an increase in complexity from simple Bitstreams to Information Packages.

It should also be added that the fact that regarding three upper Layers in the proposed model as SaaS in no way contradict the NIST definition with only three layers. If a cloud software engineer or other technical person want to interpret the three upper layers in the proposed model as SaaS for system implementation purposes, that would be perfectly acceptable.

The proposed model proved to be a convenient foundation for further development. It formed the conceptual model for and application profile and an ontology. When developing the OWL ontology, one thing that quickly became apparent was its size. Even after deliberately limiting the scope by not including other ontologies such as FOAF etc, the ontology turned out to be larger than anticipated. The reason for this is almost entirely due to the choice of PREMIS for preservation metadata. PREMIS is very comprehensive, and in conversation with experts such as Tim Gollins from the National Archives of the UK and Steve Knight, the Associate Director National Digital Library at National Library of New Zealand, I have learned that very few archival institutions chose to implement the entire PREMIS data dictionary. Instead, they either implement a subset, or use their own metadata schema, which can then be mapped to PREMIS. It should, however, be said that just because the ontology is complicated does not mean that the metadata for the digital objects will be. The amount of preservation metadata elements, depend on a number of factors, such as how many elements from the PREMIS data dictionary are used and of course on the complexity of the digital objects as shown in section 9.

A final point that should be discussed is what would happen in my model if the data in the cloud archive would need to be migrated. As technology becomes obsolete, new service requirements emerge, media suffers from decay, etc, it may become necessary for archives to carry out migration of digital contents. There is no reason to think that a cloud-based system would be any different.

As mentioned earlier, a benefit of the layered model presented in this research

is that it is an abstracted model. Any migration would need to take this into account. In other words, any migration happening in one layer should not affect above layers that rely on services from below. Using the layered model as a framework, migration could be expected to happen in any of the four layers. When it comes to the migration of raw data, this would happen in the PaaS Layer. This type of migration can be either what is known in OAIS terms as a *Refreshment* (storage media is replaced by a media instance of the same type by copying the bits on the medium used to hold data to another) or a *Replication* (A Digital Migration where there is no change to the Packaging Information, the Content Information and the PDI, but the media may be different). Both these types of migration would be possible in the PaaS Layer, and would have no impact on functionality in higher layers. However, when it comes to Repackaging or Transformation, things become more complicated, because here there is an actual change in the data. The two most important points to keep in mind when performing these two types of migration in the PaaS Layer are: 1) If there is any change to the URI of any object, this should be updated in the Information Packages sent to the archive application in the Interaction Layer. This task should be carried out by the Preservation Service. 2) Any changes to metadata or Digital Objects should be recorded as part of the Preservation Description Information. Finally, as with all migrations, it must be stressed that due auditing and quality control should be performed to ensure that nothing has gone wrong.

Migration can also be performed in other layers. The most common type of migration that can be expected to occur is a software migration from one version of a SaaS system to another (or to another SaaS system with similar functionality). Such a migration could potentially affect the type of Digital Object that is produced. For example, a newer version of a word processing application may store contents as “.docx”, where the previous version was using “.doc”. When such a software migration takes place, it must be recorded as it can have a big impact on preservation metadata (e.g. Representation Information). In such cases, the Creating Application would need to re-register with the Preservation Service, and it would be up to the Preservation Service to ensure that adequate Representation Metadata etc. about the new format is recorded. Only when this is available can the Creating Application be allowed to save

contents in a new format.

When it comes to migration in the Preservation Layer, this poses no problems, as long as the preservation service can provide the same service as before. The same can be said for migration in the Interaction Layer.

The author believes that using a layered model model is of benefit when it comes to migration. Abstraction of services mean that upgrades can be made invisible from a user perspective. Indeed, one of the main benefits of using the cloud is that consumers do not have to worry about what goes on in the PaaS Layer.

Finally, it is interesting to note that OAIS defines three separate attributes of a migration. The first two of these attributes should be met by an archive, irrespective of whether it is a traditional archive or one hosted in the cloud. However, the third attribute "... *full control and responsibility over all aspects of the transfer resides with the OAIS."* This aspect could be problematic in a cloud-based archive that relies on storage managed by an external provider. This is yet another example of the challenges in applying OAIS recommendations to cloud systems.

The presented research has set out to answer the initial research questions, however, there are still areas that could benefit from further exploration, but that have been omitted due to a lack of time. Most of all, it would have been desirable to have been able to build a complete system based on the proposals in this thesis to explore the extent to which automation was possible. It would also have been interesting to experiment with migration, to see whether the model of abstraction proposed was practical in a real life environment. Finally, the author would have liked to carry out testing with a bigger dataset.

14 Conclusion

This thesis started by examining the concept of cloud computing, and seeing how this concept fits in with traditional archiving models, exemplified by OAIS. To answer the first research question posed, the author attempted to map the Functional Entities in OAIS to cloud layers, out of a wish to make it possible for organisations to choose a “Service Level”, a way of deciding how to split the functional entities of an archive between in-house and cloud. This approach proved to be difficult for a number of reasons, but mainly because as explained in section 6.1, the OAIS entities do not map well to cloud layers of service. was what prompted the development of a layered model for cloud systems. This, however, was also not without difficulty. A simple split between Archive and Producer on one hand and a cloud platform for storage on the other, does not take into account the fundamental difference between the requirements of Archive and Producer. Even though they share the common goal of preserving content, a Producer is mainly interested in being able to access its data in its original format, using whichever Creating Application was originally used. An archive on the other hand needs to ensure the long term preservation of contents, so simple bit-level storage is not enough. This is where the Preservation Service comes into place as a bridge between the Producer and Archive, to collect information from the producer and cloud storage and to present it to a digital archive in such a way that it can be ingested and that all metadata needed for long-term preservation of Digital Objects is present. A model with four layers was suggested in section 7, and in order to test whether such a model was applicable or helpful, a test case was presented using the Japanese Government and the Japanese National Archives as an example. These were chosen partly because there was a real need for an improved submission process, but also because there were already plans to create a cloud solution for government agencies and offices in Japan (Unfortunately, this ambitions project has not come to fruition at the time of writing this thesis in 2013). The model was able, in broad terms, to describe the different types of information necessary for a move to the cloud to work, and was helpful in assigning responsibilities for services using the proposed layers.

Whereas a conceptual model is helpful in describing overall entities and

information types in a specific domain, it is too broad to be used in actual implementation. Because complete and correct preservation metadata is crucial for digital archives, the focus shifted to defining what metadata would be needed in a cloud archiving system, not only to define Digital Objects stored in the cloud, but also to describe system entities, as these are integral not only for when designing a system and assigning responsibility to layers, but also for provenance reasons. An approach built on the DCMI Singapore framework was used to define the metadata used in the creation of Submission Information Packages. PREMIS was used to express preservation metadata and cloud system metadata was added to create METS packages that could be understood by an archive. Based on the decisions made when implementing PREMIS with METS, it was possible to create such packages, and analysis of an example package showed that there was a high potential for the automated provision of preservation metadata, if a system of registration was used for Producing Applications. It was also shown that the proposed Preservation Service provided a large amount of the metadata in the packages.

Whereas the proposed Application Profile and model answered the research questions dealing with how a model can be developed in such a way that it integrates the requirements of both producer and digital archive when building a cloud-based digital archive and what such a system would look like, one major barrier to actual implementation was the lack of a formal semantic model and common vocabulary using a machine-readable format. This led to the development of an OWL ontology for cloud archive systems built on the LoC PREMIS ontology combined with the layered model of cloud computing, defining classes and their allowable domains, ranges and properties and providing a semantic framework allowing linking to metadata from other schemas.

The author believes that the strength of the ontology lies in the fact that it not only describes a metadata model for Information Packages, but also for the entities contributing to these packages. This is important in an environment like the cloud, where the sharing of computing resources (such as storage) is common, and where different information generating entities may not be capable of supplying Submission Packages in a format defined by an archive. Furthermore, without a common vocabulary and information model, it is difficult

to describe the different cloud entities that contribute to the creation of Information Packages in a manner that makes sense to both producers and archives, thus making interoperability difficult.

It was also shown that the proposed OWL ontology provided a powerful tool to validate Information Package metadata in RDF using SPARQL, not only identifying incorrect metadata, but also to point out the origin of that metadata to correct the problem and preventing it from reoccurring.

The ontology was used to describe a number of cloud system components, such as platform, storage and creating application together with a PREMIS version 2.1 Sample Record. In the model system, it was found that the ontology was able to describe the chosen components successfully, and that it allowed some metadata interoperability between content creating applications and the preservation service. So far, the model system has provided a proof-of-concept by showing an example information flow between system entities. In future, it would be worthwhile to create an integrated system that implements a storage controller to allow better abstraction of the Cloud Storage and a registration framework.

Acknowledgements

The writing of this thesis has been a journey that started when the author first enrolled in the Masters course at the University of Tsukuba, back in 2006. During my studies, I have received support and advice from a number of people without whom the writing of this thesis would not have been possible. I am greatly indebted to these people for their support.

First and foremost, I would like to express my deepest gratitude to my academic advisor Professor Shigeo Sugimoto, who despite a very busy schedule has always been generous with his time and excellent advice. Thanks to Professor Sugimoto, I have had the unique opportunity to experience not only Japanese culture, but also the Japanese world of archives and records management at the highest level. The Sugimoto/Nagamori Lab has always been a friendly and supportive environment, and I cherish all the friends I have made there.

I would also like to express my sincerest thanks to Associate Professor Mitsuharu Nagamori for his patience and for his skill in explaining the art of programming and of his expert guidance in the subjects of RDF, OWL and XML.

I would also like express my gratitude to my co-advisors, Associate Professor Atsuyuki Morishima and Associate Professor Tetsuo Sakaguchi for their advice, lectures and feedback at our joint lab seminars.

I am greatly indebted to the Rotary Yoneyama Memorial Foundation for their financial support and especially to my counsellor Mr Yukio Iizuka for his friendship and many kindnesses.

Special thanks goes to professor Liddy Nevile who has been a great friend and a constant source of support and advice. I am especially indebted to her for her support when I arrived in Australia with nowhere to live.

I am also indebted to Stuart Little Weibel for his company in Japan and for his professional help.

Last, but not least, I want to thank my wife Akiko Ishibashi for her love, companionship and understanding during my studies. If it had not been for her patient support, I would not have been able to complete my studies.

References

Amazon, 2011. Amazon Simple Storage Service (Amazon S3). Available at: http://aws.amazon.com/s3/ [Accessed January 14, 2011].

Askhoj, J., Nagamori, M. & Sugimoto, S., 2011. Archiving as a service: a model for the provision of shared archiving services using cloud computing. In Proceedings of the 2011 iConference. pp. 151–158.

Askhoj, J., Sugimoto, S. & Nagamori, M., 2010. Reconsidering the OAIS Reference Model for Record Management and Archiving in a Cloud Computing Environment. Available at: www.dcc.ac.uk/webfm_send/301 [Accessed April 14, 2011].

Askhoj, Jan & Sugimoto, Shigeo, 2010. A Model for the Provision of Preservation Metadata as a Service. In THE 2010 CISAP Colloquium On Digital Library Research. Taipei, Taiwan.

Askhoj, Jan, Sugimoto, Shigeo & Nagamori, M., 2007. Constructing a records archiving system using off-the-shelf tools - A light weight approach. In 7th International Web Archiving Workshop 2007. Vancouver, Canada. Available at: http://iwaw.europarchive.org/07/index.html [Accessed January 19, 2013].

Badger, L. et al., 2012. Cloud Computing Synopsis and Recommendations. Recommendations of the National Institute of Standards and Technology. Available at: http://csrc.nist.gov/publications/PubsSPs.html#800-146 [Accessed January 19, 2013].

Beagrie, N. & Jones, M., 2008. Introduction - Definitions and Concepts. In Preservation Management of Digital Materials: The Handbook. Heslington, York: Digital Preservation Coalition. Available at:

http://www.dpconline.org/advice/preservationhandbook [Accessed January 20, 2013].

Beckett, D. & Berners-Lee, T., 2008. Turtle-terse RDF triple language. W3C Team Submission, 14.

Cachin, C., Keidar, I. & Shraer, A., 2009. Trusting the cloud. ACM SIGACT News, 40(2), pp.81–86.

Caplan, P. 2009. Understanding PREMIS : an overview of the PREMIS Data Dictionary for Preservation Metadata. Library of Congress. Available at: http://www.loc.gov/standards/premis/understanding-premis.pdf [Accessed November 1, 2012].

Caplan, P., Kehoe, W. & Pawletko, J., 2010. Towards Interoperable Preservation Repositories (TIPR). In Proceedings of the 2010 Roadmap for Digital Preservation Interoperability Framework Workshop, 1–4. Gaithersburg, Maryland: ACM.

CCSDS Secretariat, 2002. Reference Model for an Open Archival Information System (OAIS). In Blue Book. Consultative Committee for Space Data Systems. Available at: public.ccsds.org/publications/archive/650x0b1.PDF [Accessed November 2, 2011].

CCSDS Secretariat 2, 2002. Reference Model for an Open Archival Information System (OAIS) version 2. Consultative Committee for Space Data Systems. Available at: http://public.ccsds.org/publications/archive/650x0m2.pdf [Accessed March 3, 2013].

Chan, T., 2009. Japan to build massive cloud infrastructure for e-government. Green Telecom. Available at:

http://www.greentelecomlive.com/2009/05/13/japan-to-build-massive-cloud-infrastructure-for-e-government/.

Chellappa, R., 1997. Intermediaries in Cloud-Computing: A New Computing Paradigm. Available at:

http://meetings2.informs.org/Dallas97/TALKS/MD19.html [Accessed January 19, 2013].

Chia-Chi Teng et al., A medical image archive solution in the cloud. Software Engineering and Service Sciences (ICSESS), 2010 IEEE International Conference on, pp.431–434.

Clark, K.G., Feigenbaum, L. & Torres, E., 2008. SPARQL protocol for RDF. World Wide Web Consortium (W3C) Recommendation.

Cong Wang et al., 2012. Toward Secure and Dependable Storage Services in Cloud Computing. Services Computing, IEEE Transactions on, 5(2), pp.220–

232.

Daniels, M.F., 1984. Introduction to Archival Terminology. In A Modern Archives Reader: Basic Readings on Archival Theory and Practice. National Archives Trust Fund Board, pp. 336–342. Available at:

http://www.archives.gov/research/alic/reference/archives-resources/terminology.html [Accessed January 19, 2013].

Dappert, Angela, 2008. The PREMIS Data Dictionary:#Information you need to know for preserving digital documents .

DoD, 2007. Design Criteria Standard for Electronic Records Management Software Applications, Arlington, VA: Department of Defence. Available at: www.dtic.mil/whs/directives/corres/pdf/501502std.pdf [Accessed January 20, 2013].

Enders, M., 2010. A METS Based Information Package for Long Term Accessibility of Web Archives. In Proceedings of the 7th International Conference on Preservation of Digital Objects. iPRES 2010. Vienna, Austria, p. 8. Available at: www.ifs.tuwien.ac.at/dp/ipres2010/papers/enders-70.pdf [Accessed June 10, 2011].

FCLA, 2011. PREMIS in METS Toolbox. Pimtools, V1.0.1. Available at: http://pim.fcla.edu/. [Accessed Aug 15, 2012].

Gantz, J. & Reinsel, D., 2010. The Digital Universe Decade – Are You Ready? IDC iView. Available at: http://idcdocserv.com/925.

Gartner, R., 2002. METS: Metadata Encoding and Transmission Standard. JISC Techwatch Report TSW, pp.02–05.

Gartner, R., 2004. PREMIS - Preservation Metadata Implementation Strategies Update 2: Core Elements for Metadata to Support Digital Preservation. RLG Diginews, 8(6), pp.2007–02.

Giaretta, D. et al., 2005. Representation information for interoperability now and with the future. In Local to Global Data Interoperability - Challenges and Technologies, 2005. pp. 42–46.

Gorgan, M. & Rushay, F., 2009. Backyard Botanicals Metadata Dictionary and Crosswalk. Available at:

http://dspaceslis.kent.edu:8080/jspui/handle/123456789/1119.

Guenther, R., 2009. Understanding and Implementing the PREMIS Data Dictionary for Preservation Metadata. Available at:

http://www.digitalpreservation.gov/news/events/ndiipp_meetings/ndiipp09/docs/June26/premis-ndiipp-20090626.ppt.

Hedstrom, M., 1997. Digital preservation: a time bomb for digital libraries. Computers and the Humanities, 31(3), pp.189–202.

Huth, A. & Cebula, J., 2011. The Basics of Cloud Computing. Available at: http://www.butp.org/nscc/Pdf/USCERT-CloudComputingHuthCebula.pdf [Accessed January 20, 2013].

ISO, 2001. 15489-1: 2001. Information and Documentation: Records Management Part, 1 - General.

Jain, L. & Bhardwaj, S., 2010. Enterprise Cloud Computing: Key Considerations for Adoption.

Jia-sun, H.E., 2001. Life Cycle of Electronic Records. Journal of Zhejiang University (Humanities and Social Sciences), p.04.

Jones, M. & Beagrie, N., In Digital Preservation Handbook.

Koga, T., 2010. Recent development of the government information policy in Japan. Government Information and Official Publications Section (GIOPS) Newsletter, 8, pp.8–11.

Krangel, E., 2009. Larry Ellison: Someone Explain To Me This “Cloud Computing” Thing My Company Is Committing To. Business Insider. Available at: http://www.businessinsider.com/2008/9/larry-ellison-someone-explain-to-me-this-cloud-computing-thing-my-company-is-committing-to-orcl- [Accessed January 19, 2013].

Krigsman, M., 2008. MediaMax / The Linkup: When the cloud fails. ZDNet. Available at: http://www.zdnet.com/blog/projectfailures/mediamax-the-linkup-when-the-cloud-fails/999 [Accessed January 20, 2013].

Kulovits, H. et al., 2008. Plato: A Preservation Planning Tool Integrating Preservation Action Services. Research and Advanced Technology for Digital

Libraries, pp.413–414.

Leavitt, N., 2010. Is Cloud Computing Really Ready for Prime Time? Computer 42, January(1), pp.15–20.

Lenk, A. et al., 2009. What's inside the Cloud? An architectural map of the Cloud landscape. In Software Engineering Challenges of Cloud Computing, 2009. CLOUD'09. ICSE Workshop on. pp. 23–31.

McGuinness, D.L. & Van Harmelen, F., 2004. OWL Web Ontology Language Overview. Available at: http://www.w3.org/TR/owl2-overview/.

McLeod, J., 2002. Managing electronic records. Records Management Journal, 12(3).

Mell, P. & Grance, T., 2009. The NIST definition of cloud computing. National Institute of Standards and Technology, 53(6).

Miles, A. et al., 2005. Skos Core: Simple Knowledge Organisation for the Web. International Conference on Dublin Core and Metadata Applications, p.3.

Miri, J. & Mintz Testa, B., 2011. Cloud Computing. Public CIO, 2011(2). Available at: http://www.njslom.org/presentations/Cloud_Special_Report.pdf [Accessed January 19, 2013].

Mohamed, A., 2009. A history of cloud computing. Computer Weekly. Available at: http://www.computerweekly.com/feature/A-history-of-cloud-computing [Accessed January 19, 2013].

Muniswamy-Reddy, K.-K., Macko, P. & Seltzer, M., 2010. Provenance for the cloud. In Proceedings of the 8th USENIX conference on File and storage technologies. San Jose, California: USENIX Association, pp. 15–14.

NARA, 2004. Electronic Records Archives ERA Lifecycle. Available at: http://www.archives.gov/era/pdf/era-life-cycle.pdf.

National Archives of Japan, 2007. National Archives Law.

Nguyen, Q.L. & Lake, A., 2011. Content Server System Architecture for Providing Differentiated Levels of Service in a Digital Preservation Cloud. Cloud Computing (CLOUD), 2011 IEEE International Conference on, pp.557–564.

Office of the Victorian Privacy Commissioner, 2011. Privacy Victoria - Cloud

Computing. Available at:

https://www.privacy.vic.gov.au/privacy/web2.nsf/files/cloud-computing [Accessed January 20, 2013].

Okamoto, S., 2010. New Developments in Managing Records in Japan - The Establishment, Direction and Structure of the Archive Law. Presentation at the Canadian Embassy, Tokyo, Japan.

Peyrard, S.X., 2011. Public Workspace for PREMIS OWL Ontology. Available at: http://premisontologypublic.pbworks.com/w/page/45987067/FrontPage.

Premis Editorial Committee, 2011. PREMIS Data Dictionary for Preservation Metadata Version 2.1. PREMIS Editorial Committe. Available at: www.loc.gov/standards/premis/v2/premis-2-1.pdf.

PREMIS in METS Working Group, 2008. Guidelines for using PREMIS with METS for exchange, Revised June.

PROV, 2003. VERS Standard 99/007: Management of Electronic Records 2nd ed., Melbourne, Victoria: Public Records Office of Victoria.

Robles, M. & Langemo, M., 1999. The Fundamentals of Records Management 1st ed., Office Systems.

Salesforce.com, 2012. Cloud Computing - The Complete History of Cloud Computing - Salesforce.com UK. Available at:

http://www.salesforce.com/uk/socialsuccess/cloud-computing/the-complete-history-of-cloud-computing.jsp [Accessed January 19, 2013].

Schuller, S., 2008. SaaS, PaaS, IaaS Differences, Cloud Diagram & Examples | SaaS Blogs. SaaSBlogs. Available at: http://www.saasblogs.com/saas/demystifying-the-cloud-where-do-saas-paas-and-other-acronyms-fit-in/ [Accessed January 19, 2013].

Securities and Exchange Commission, 2003. Final Rule: Retention of Records Relevant to Audits and Reviews. Available at: http://www.sec.gov/rules/final/33-8180.htm [Accessed January 20, 2013].

Sierman, B., 2012. OAIS 2012 update. *Digital Preservation Seeds*. Available at: http://digitalpreservation.nl/seeds/oais-2012-update/ [Accessed March 3, 2013].

Storage Networking Industry Association, 2012. Cloud Data Management Interface (CDMI). SNIA Technical Position. Available at:

http://snia.org/sites/default/files/CDMI%20v1.0.2.pdf [Accessed January 19, 2012].

Story, L. & Stone, B., 2007. Facebook Retreats on Online Tracking. The New York Times. Available at:

http://www.nytimes.com/2007/11/30/technology/30face.html?_r=1& [Accessed January 13, 2013].

Sugimoto, Shigeo, 2007. Ensuring the Preservation and Use of Electronic Records. Available at: http://www.archives.go.jp/english/news/pdf/sugimoto.pdf [Accessed January 1, 2011].

The OCLC/RLG Working Group on Preservation Metadata, 2002. Preservation Metadata and the OAIS Information Model - A Metadata Framework to Support the Preservation of Digital Objects. Available at:

http://www.oclc.org/research/activities/past/orprojects/pmwg/pm_framework.pdf [Accessed January 14, 2011].

UK Public Record Office, 2002. Requirements for electronic records management systems 2: Metadata Standard. Available at:

www.nationalarchives.gov.uk/documents/metadatafinal.pdf [Accessed June 11, 2010].

Vaquero, L.M. et al., 2008. A break in the clouds: towards a cloud definition. ACM SIGCOMM Computer Communication Review, 39(1), pp.50–55.

Vermaaten, S., 2010. A Checklist and a Case for Documenting PREMIS METS Decisions in a METS Profile. D-Lib Magazine, October(16).

W3C, 2009. OWL 2 Web Ontology Language Document Overview. W3C OWL Working Group. Available at: http://www.w3.org/TR/owl2-overview [Accessed March 12, 2012].

Wetteroth, D., 2001. OSI reference model for telecommunications, McGraw-Hill Professional.

Woodyard-Robinson, D., 2007. Implementing the PREMIS Data Dictionary: a S

urvey of Approaches. Library of Congress, 29:2009 (29). Available at: www.loc.gov/standards/premis/implementation-report-woodyard.pdf. [Accessed January 14, 2011].

Yamato, Y. et al., 2012. Survey of Public IaaS Cloud Computing API. IEEJ Transactions on Electronics, Information and Systems, 132(1), pp.173–180.

Youseff, L., Butrico, M. & Da Silva, D., 2009. Toward a unified ontology of cloud computing. In Grid Computing Environments Workshop, 2008. GCE'08. pp. 1–10.

Zend, Simple Cloud API Reference. Available at: http://simplecloud.org/api [Accessed January 14, 2011].

List of Publications

1. Jan Askhoj, Shigeo Sugimoto, Mitsuharu Nagamori, "Preserving records in the cloud", Records Management Journal, Vol. 21 Issue 3, pp.175-187, 2011

2. Jan Askhoj, Shigeo Sugimoto, and Mitsuharu Nagamori, “A metadata framework for cloud-based digital archives using METS with PREMIS”, Proceedings of the 13th International Conference on Asia-Pacific Digital Libraries: for cultural heritage, knowledge dissemination, and future creation (ICADL'11), Lecture Notes in Computer Science 7008, Springer-Verlag, , pp. 118-127, 2011

3. Jan Askhoj, Mitsuharu Nagamori, and Shigeo Sugimoto, “Archiving as a service: a model for the provision of shared archiving services using cloud computing”, Proceedings of the 2011 iConference (iConference '11), ACM, New York, NY, USA, pp. l5l-158, 2011

4. Jan Askhoj, Shigeo Sugimoto, and Mitsuharu Nagamori, “An Ontology for Automated Cloud Archiving Systems”, International Workshop on Global Collaboration of Information Schools (WIS), Taipei, 2012

5. Jan Askhoj, Shigeo Sugimoto, “Reconsidering the OAIS Reference Model for Record Management and Archiving in a Cloud Computing Environment”, 6th International Digital Curation Conference (DCC), Chicago, Poster, 2010.

6. Jan Askhoj, Shigeo Sugimoto, “A Model for the Provision of Preservation Metadata as a Service”, The 2010 CISAP Colloquium on Digital Library Research, Taipei, 2010

Appendix 1. Application Profile using PREMIS with METS in Spreadsheet format

(Usage example in XML and rationale have been omitted due to lack of space)

PREMIS/METS	CONTAINER	MAND.	REPEAT	ORIGIN	Element	Explanation	Example 1 - PDF file	Example 2 - Email (top)
					Root element			
METS Root		M	N	P(imp)	OBJID	Is the primary identifier assigned to the METS object as a whole. Although this attribute is not required, it is strongly recommended. This identifier is used to tag the entire METS object to external systems, in contrast with the ID identifier	l3j44klj23lj4l2	df0g80f8gdgdg
METS Root		M	N	P(imp)	ID	This attribute uniquely identifies the element with which the root element is associated within the METS document, and which would allow the element to be referenced unambiguously from another element or document via an IDREF or an XPT	ROO1	ROO1
METS Root		M	N	P(reg)	XML Definition	Not part of METS as such	xml version="1.0" encoding="UTF-8”	xml version="1.0" encoding="UTF-8”

METS Root		M	R	P(imp)	Namespace Definitions	Not part of METS as such, Mets, PREMIS, Mods namespaces. Similar to the METS 'PROFILE' field.	http://www.loc.gov/METS/ mets http://www.loc.gov/mods/v3/ mods http://www.loc.gov/standards/premis/v2/	http://www.loc.gov/METS/ mets http://www.loc.gov/mods/v3/ mods http://www.loc.gov/standards/premis/v2/
METS Root		M	N	B(exp)	Type	Is the DO a container or an object	Object	Container
METS		O			Other optional elements:	LABEL, TYPE, PROFILE		
					Header (creator, date, etc.) - metsHdr			
METS Header		M	N	P(imp)	ID	An attribute that uniquely identifies the <metsHdr> element which would allow the element to be referenced unambiguously from another element or document via an IDREF or an XPTR.	HDR1	HDR1
METS Header		M	N	P(imp)	Created Date	Date and time of IP creation	2011-03-31T12:00:00+09:00	2011-04-31T12:00:00+09:00
METS Header		M	N	P(imp)	Last Moderated Date	Date and time of IP last moderation	2011-03-31T12:00:00+09:00	2011-05-02T15:34:00+09:00
METS Header		M	N	P(reg)	Preservation Service Name	Name of used preservation service	Foo Preservation Service	Foo Preservation Service

METS Header		M	N	P(reg)	Preservation Service URI	URL of used preservation service	http://foo.org/preservation_service	http://foo.org/preservation_service
METS Header		M	N	P(imp)	Recordstatus	Specifies the status of the METS document. It is used for internal processing purposes	Completed	Completed
METS		O	R	P(imp)	<mets:agent>	The agent element <agent> provides for various parties and their roles with respect to the METS record to be documented		
METS		O	R	P(imp)	ID	Identifies the <agent> element which allows the element to be referenced unambiguously from another element or document via an IDREF or an XPTR	ID: http://foo.org/preservation_service	ID: http://foo.org/preservation_service
METS		O	R	B(reg)/P(imp)	Role	Values are CREATOR, EDITOR, ARCHIVIST, PRESERVATION, DISSEMINATOR, CUSTODIAN, IPOWNER, OTHER	Preservation: Foo Preservation Service	Preservation: Foo Preservation Service
METS		O			Other optional elements:	ADMID, OTHERROLE, TYPE, OTHERTYPE		
METS		O			<Alternative Identifiers>	Type		
					Digital Object Info - fileSec			
METS File		M	N	P(imp)	ID	A unique identifier for the <fileSec>, thus allowing it to be referenced via an IDREF elsewhere in the METS document	FID1	FID2
METS File		M	N	B(reg)	DO Location	Location identifier of Digital Object. All need to be in one location.	/remote/path/	/remote/path/

METS File		M	R	B(exp)	DO Unique Identifier	Unique identifier of Digital Object. Must be unique within Location	x.pdf	a (dummy file)
METS File		M	R	B(exp)	MD Unique Identifier	Unique identifier of Metadata. Must be unique within Location	x-MD.xml	a-MD.xml
METS	C				<file>			
		O			Other optional elements:	ID, MIMETYPE, SEQ, SIZE, CREATED, CHECKSUM, CHECKSUMTYPE, OWNERID, ADMID, DMDID, GROUPID, USE		
METS	C				<fileGrp>			
METS		O			Other optional elements:	ID (grp), VERSDATE, ADMID, USE		
					Descriptive metadata (DC/MODS) - dmdSec - stored externally and pointed to (mdRef).			
METS Dmd		M	R	P(imp)	ID	Provides a unique, internal name for each <dmdSec> element	DMD1	DMD1
METS		O			Other optional elements:	GROUPID, ADMID, CREATED, STATUS		
METS	C	O			EXTERNAL DESCRIPTIVE METADATA	ID, MIMETYPE, LABEL, LOCTYPE, OTHERLOCTYPE, MDTYPE, OTHERMDTYPE		

METS	C	O			INTERNAL DESCRIPTIVE METADATA	ID, MIMETYPE, LABEL, MDTYPE, OTHERMDTYPE		
Original MD		M	N	MD	Identifier	See DC	External System Identifier xxx	External System Identifier xxx
Original MD		M	N	MD	Title	See DC	Sugimoto Lab Research Plan	Email reply to Director of NHK
Original MD		M	N	MD	Creator	See DC	Mitsuharu Nagamori	Shigeo Sugimoto
Original MD		O	N	MD	Subject	See DC	Library and Information Science	Research Budget
Original MD		O	N	MD	Description	See DC	This document is about…	This document is about…
Original MD		O	N	MD	Publisher	See DC	Tsukuba University	Tsukuba University
Original MD		O	N	MD	Contributor	See DC	Shigeo Sugimoto	Shigeo Sugimoto
Original MD		M	N	MD	Date	See DC	23-01-2011	23-01-2011
Original MD		O	N	MD	Type	See DC	HTML Document	Email with attachment
Original MD		O	N	MD	Format	See DC	Text	Email
Original MD		O	N	MD	Source	See DC	Google Docs	Simplemailer
Original MD		O	N	MD	Language	See DC	English	English
Original MD		O	N	MD	Relation	See DC	None	None
Original MD		O	N	MD	Coverage	See DC	Japan	Japan

Original MD		O	N	MD	Rights	See DC	Open	Open
					Administrative metadata 1 (general) amdSec			
METS Adm		M	R	P(imp)	ID	This attribute uniquely identifies the element with which the root element is associated within the METS document, and which would allow the element to be referenced unambiguously from another element or document via an IDREF or an XPT	AMD1	AMD1
METS Adm		M	N	B(reg)	Business System Name	Name of originating business system	Sugimoto Lab Research CMS	Tsukuba University Tulips Mail
METS Adm		M	N	B(reg)	Business System URI	URI of originating business system	https://docs.google.com/?pli=slrcms	
https://www.tulipsmail/access						C	O	
<techMD>Technical MetadataPartly Covered by PREMIS sectionPartly Covered by PREMIS sectionMETS		O			Other optional elements:	ID, GROUPID, ADMID, CREATED, STATUS		
METS	C	O	R		<rightsMD>	Rights Metadata		

METS	C	O	R		<sourceMD>	Source Metadata		
METS	C	O	R		<digiprovMD > See below	Digital Providence MD		
					Administrative metadata 2 (PREMIS Metadata) amdSec.			
				B(reg)	act			
				B(reg)	Copyright Juristiction			
1.1	C	M	R	P(imp)	objectIdentifier	Combination of type and value should be globally unique.		
1.1.1	Con Voc	M	N	P(reg)	objectIdentifierType	Use identifier automatically created by the repository as the primary identifier. Implement a handle system for use as the primary identifier within the repository, and use this value: handle. The handle syntax should include a substring for the collection and sequential suffixes to indicate structural or derivative relationships among objects (e.g., yale.dp/ydc2593-001 where dp indicates the repository, ydc indicates the collection, and 001 indicates that the object is the original object submitted to the repository).	UUID	UUID
1.1.2	None	M	N	P(imp)	objectIdentifierValue		see Digital Object Info - fileSec	see Digital Object Info - fileSec
1.2	Con Voc	M	N	P(imp)	objectCatego	Use these values: Representation, File,	File	Representation

					ry	Bitstream.		(container)
1.3	C	O	R		preservation Level	Use a controlled vocabulary. Assign at the repository or institution level rather than at the object level. Determine according to one or more of the following factors: cultural value, uniqueness, preservability, costs, etc. (e.g., Rutgers Libraries uses two different sets of values: (1) high, medium, low, or none and (2) full, or bitstream). Associate each value for preservationLevel with a profile that designates which semantic units are mandatory and how their values should be recorded. Profiles for higher preservation levels would have more stringent requirements. Requires policy decision by DPC.		
1.3.1	Con Voc	M	N	P(reg)	preservation LevelValue		full	full
1.3.2	Con Voc	O	N	P(reg)	preservation LevelRole	A value indicating the context in which a set of preservation options is applicable	requirement	requirement
1.3.3	None	O	R	P(reg)	preservation LevelRationale	The reason a particular preservationLevelValue was applied to the object	nil	Format soon to be discontinued
1.3.4	None	O	N	P(reg)	preservation LevelDateAssigned		2010-08-01T09:08:44-03:00	2010-08-01T09:08:44-03:00

1.4	C	O	R		significantProperties	the repository can decide that for all PDF files, only the content need be preserved. In other cases, for example, for media art, the significant properties may be unique to each individual object Where values are unique, they must be supplied by the submitter or provided by the curatorial staff of the repository.		
1.4.1	None	O	N	P(reg)	significantPropertiesType	The aspect, facet, or attribute of an object about which significant properties are being described	content	content
1.4.2	None	O	N	P(reg)	significantPropertiesValue		non-editable	non-editable
1.4.3	C	O	R		significantPropertiesExtension			
1.5	C	M	R		objectCharacteristics	Used to record technical properties. Format-specific properties are out of scope for PREMIS.		
1.5.1	Non-negative integer	M	N	B(reg)	compositionLevel	Supply value even when object is uncompressed and unencrpyted, e.g., assign 0 for base level, 1 for compressed file, 2 for compressed and encrypted file. Should be supplied by repository (e.g. at registration)	0	0
1.5.2	C	O	R		fixity	Investigate cost to generate and maintain message digests calculated by one or more algorithms. The PREMIS Data Dictionary recommends using two or more message digests calculated by different algorithms. Requires policy decision by DPC.		

1.5.2.1	Con Voc	M	N	B(reg)	messageDigestAlgorithm	Use registry. Until a global registry becomes available, create and maintain a local registry of encryption algorithms by type: hash algorithms, symmetric algorithms, asymmetric algorithms. Use entries in the hash algorithm registry to populate this semantic unit. Requires policy decision by DPC.	MD5	MD5
1.5.2.2	None	M	N	P(imp)	messageDigest	Definition The output of the message digest algorithm.	7868792365	232352422
1.5.2.3	None	O	N	P(reg)	messageDigestOriginator	Record in Object entity or in Event entity.	http://foo.org/preservation_service	http://foo.org/preservation_service
1.5.3	Non-negative integer	O	N	P(imp)	size	Record value obtained from format validation performed during ingest. Use bytes as unit of measurement.	135 kb	90 kb
1.5.4	C	M	R		format	Record by value in formatDesignation or by reference in formatRegistry. Requires policy decision by DPC.		
1.5.4.1	C	O	N		formatDesignation	Use a controlled vocabulary.		
1.5.4.1.1	Con Voc	M	N	Ext/P(reg)	formatName		application/pdf	multipart/alternative (email)
1.5.4.1.2	Con Voc	O	N	Ext/P(reg)	formatVersion		PDF/A-1a	Default
1.5.4.2	C	O	N		formatRegistry	Identify appropriate global registry or develop local registry. See Usage Notes for creatingApplication.	info:gdfr/fred/p/pdf/15	info:gdfr/fred/p/email/12
1.5.4.2.1	None	M	N	P(reg)	formatRegistryName	Use formal name, local name, or URI consistently. Requires policy decision by DPC.	http://nationalarchives.gov.uk/PR	http://nationalarchives.gov.uk/PR

							ONOM/	ONOM/
1.5.4.2.2	None	M	N	P(reg)	formatRegistryKey	Requires policy decision by DPC.	info:gdfr/fred/p/pdf/15	info:gdfr/fred/p/email/12
1.5.4.2.3	Con Voc	O	N	P(reg)	formatRegistryRole	Requires policy decision by DPC.	Validation profile	Validation profile
1.5.4.3	None	O	R	P(reg)	formatNote		nil	tentative identification
1.5.5	C	O	R		creatingApplication	Use registry. Until a global format registry becomes available, create and maintain a local registry of file formats with information, by function (create, render, identify, validate, etc.), on compatible software. For each application, provide a standard name, version, vendor, and system requirements. Use this registry to populate creatingApplication. Requires policy decision by DPC.		
1.5.5.1	Con Voc	O	N	B(reg)	creatingApplicationName		Google Docs	Simple-mailer
1.5.5.2	None	O	N	B(reg)	creatingApplicationVersion		nil	3.4
1.5.5.3	ISO 8601	O	N	P(imp)	dateCreatedByApplication	Express dates in the extended format with hyphens. Express times in UTC (Coordinated Universal Time) with the UTC designator ("Z").	2010-08-01T09:08:44-03:00	2011-03-01T09:08:44-03:00
1.5.5.4	C	O	R		creatingApplicationExtension	Creating application information using semantic units defined external to PREMIS		

1.5.6	C	O	R		inhibitors	Inhibitors may not be detected when a file is parsed. If applicable, require in Submission Information Package (SIP).		
1.5.6.1	Con Voc	M	N	B(reg)	inhibitorType	Use registry. Until a global registry becomes available, create and maintain a local registry of encryption algorithms by type: hash algorithms, symmetric algorithms, asymmetric algorithms. If encryption is used, use entries in the symmetric algorithm subregistry to populate this semantic unit. If password protection is used, use this value: password protection. Requires policy decision by DPC.	nil	PGP
1.5.6.2	Con Voc	O	R	B(reg)	inhibitorTarget	Use a controlled vocabulary.	nil	All content
1.5.6.3	None	O	N	B(exp)	inhibitorKey	Integrate storage of keys with university-wide Public Key Infrastructure (PKI).	nil	098543jfgh987jexs
1.5.7	C	O	R		objectCharacteristicsExtension			
1.6	None	O	N	DO	originalName	Use the file name designated in the Submission Information Package (SIP).	Sugimoto Lab Research Plan	Email reply to Director of NHK
1.7	C	M	R		storage			
1.7.1	C	O	N		contentLocation	If a handle system is implemented, contentLocation is implicit in objectIdentifier.		
1.7.1.1	Con Voc	M	N	Ext	contentLocationType	If a handle system is implemented, contentLocationType is identical to objectIdentifierType.	Cloud Platform	Cloud Platform

1.7.1.2	None	M	N	Ext	contentLocationValue	If a handle system is implemented, the information needed to resolve handles to file locations is implicit in the handle system. Documentation for the handle system should be stored by the repository but does not need to be recorded in the metadata for each object.	Archive Cloudstore X	Archive Cloudstore X
1.7.2	Con Voc	O	N	Ext	storageMedium	Implicit. Determined by repository architecture.	Archive Cloudstore X	Archive Cloudstore X
1.8	C	O	R		environment	Record information on functions supported by hardware and software in a registry, not in the metadata for each object. See Usage Notes for creatingApplication. Record information on dependent files in the metadata for each object.		
1.8.1	Con Voc	O	N	P(reg)	environmentCharacteristic	An assessment of the extent to which the described environment supports its purpose. This value could be supplied by the submitter or by the repository. If environment software and hardware information is obtained from an environments registry, environmentCharacteristic might also be obtained from the registry.	Unspecified	Known to work
1.8.2	Con Voc	O	R	P(reg)	environmentPurpose	Different environments can support different uses of objects. For example, the environment needed to edit and modify a file can be quite different than the environment needed to render it	nil	Render
1.8.3	None	O	R	P(reg)	environmentNote	There may be a need to give a textual description of the environment for additional	nil	Needs Word 97 or above

						explanation		
1.8.4	C	O	R		dependency	Use for non-executable components of an object, e.g., a font, style sheet, or schema. Do not use for software or hardware.		
1.8.4.1	None	O	R	P(reg)	dependencyName	A designation for a component or associated file needed by the representation or file	nil	Japanese Unicode plugin needed
1.8.4.2	C	O	R		dependencyIdentifier	Use objectIdentifier of dependent object.		
1.8.4.2.1	Con Voc	M	N	P(reg)	dependencyIdentifierType	A designation of the domain in which the identifier of the dependent resource is unique	URI	URI
1.8.4.2.2	None	M	N	B(exp)	dependencyIdentifierValue	Used to identify dependent object	http://foo.bar/1	http://foo.bar/2
1.8.5	C	O	R		software	Software can be inferred from creatingApplication and does not need to recorded in the metadata for each object. See Usage Notes for creatingApplication.		
1.8.5.1	None	M	N	B(reg)	swName		Google Docs	Simplemailer
1.8.5.2	None	O	N	B(reg)	swVersion		3.4	1.21
1.8.5.3	Con Voc	M	N	B(reg)	swType		Office Software	Office Software
1.8.5.4	None	O	R	B(reg)	swOtherInformation		SaaS Software	SaaS Software
1.8.5.5	None	O	R	B(reg)	swDependency		nil	nil

1.8.6	C	O	R		hardware	Hardware can be inferred from creatingApplication and does not need to be recorded in the metadata for each object. See Usage Notes for creatingApplication.		
1.8.6.1	None	M	N		hwName		Not Applicable	Not Applicable
1.8.6.2	Con Voc	M	N		hwType		Not Applicable	Not Applicable
1.8.6.3	None	O	R		hwOtherInformation		Not Applicable	Not Applicable
1.8.7	C	O	R		environmentExtension			
1.9	C	O	R		signatureInformation	Do not use to record information about digital signatures that authenticate agents; use the Event Entity instead.		
1.9.1	C	O	R		signature			
1.9.1.1	Con Voc	M	N	B(reg)	signatureEncoding	Use registry.	Base64	Base64
1.9.1.2	None	O	N	B(reg)	signer	Use the name provided in the Submission Information Package (SIP).	Sugimoto Lab Research CMS	Tsukuba University Tulips Mail
1.9.1.3	Con Voc	M	N	B(reg)	signatureMethod	Use registry. Create and maintain a local registry of encryption algorithms by type: hash algorithms, symmetric algorithms, asymmetric algorithms. Use entries in the asymmetric algorithm registry and hash algorithm registry to populate signatureMethod. Record the encryption algorithm for the signature first (the asymmetric algorithm), followed by a hyphen, followed by the hash algorithm, e.g., DSA-SHA1. Requires policy decision by DPC.	DSA-SHA1	DSA-SHA1

1.9.1.4	None	M	N	B(exp)	signatureValue		7JaYztgt4	987sdf9YTT
1.9.1.5	None	M	N	P(reg)	signatureValidationRules	May be a pointer to external documentation. See <http://www.w3.org/TR/xmldsig-core/>. Requires policy decision by DPC.	Not Used	Not Used
1.9.1.6	None	O	R	P(reg)	signatureProperties	Define suitably granular structure for time of signature generation, serial number of cryptographic hardware used, etc., as needed. Requires policy decision by DPC.	Not Used	Not Used
1.9.1.7	C	O	N		keyInformation			
1.9.2	C	O	R		signatureInformationExtension			
1.10	C	O	R		relationship	Record all relevant relationships.		
1.10.1	Con Voc	M	N	B(reg)	relationshipType	Develop a local controlled vocabulary of relationships.	nil	structural
1.10.2	Con Voc	M	N	B(reg)	relationshipSubType	Include in a local controlled vocabulary of relationships.	nil	includes
1.10.3	C	M	R		relatedObjectIdentification	Use objectIdentifier of related resource.		
1.10.3.1	Con Voc	M	N	B(reg)	relatedObjectIdentifierType		nil	XML Structure
1.10.3.2	None	M	N	B(exp)	relatedObjectIdentifierValue		nil	a-MD.xml b.html b-MD.xml c.html c-MD.xml
1.10.3.3	None	O	N	B(exp)	relatedObjectSequence	If there is only one related object, assign the value 0 (zero). If objects are unordered, assign each the same value.	nil	nil

1.10.4	C	O	R		relatedEventIdentification	Use eventIdentifier of related event.		
1.10.4.1	Con Voc	M	N	P(reg)	relatedEventIdentifierType		1 (e.g. Package Creation)	nil
1.10.4.2	Con Voc	M	N	E	relatedEventIdentifierValue		nil	nil
1.10.4.3	None	O	N	E	relatedEventSequence	If there is only one related event, assign the value 0 (zero). If events are unordered, assign each the same value.	nil	nil
1.11	C	O	R		linkingEventIdentifier	Use eventIdentifier of linking event.		
1.11.1	Con Voc	M	N	P(reg)	linkingEventIdentifierType		UUID	UUID
1.11.2	Con Voc	M	N	E	linkingEventIdentifierValue		05y50321-6d7b-4291-89ag	987ss-werb6-jsdb6-73456
1.12	C	O	R		linkingIntellectualEntityIdentifier	Use to identify an object whose content is related to the object designated by the objectIdentifier. Optionally, use to identify a metadata record for the object designated by the objectIdentifier. Requires policy decision by DPC.	Harvesting	Harvesting
1.12.1	Con Voc	M	N	B(reg)	linkingIntellectualEntityIdentifierType		Collection	Collection
1.12.2	None	M	N	B(exp)	linkingIntellectualEntityIdentifierValue		Nil	Tsukuba University Collection

1.13	C	O	R		linkingRightsStatementIdentifier			
1.13.1	Con Voc	M	N	B(reg)	linkingRightsStatementIdentifierType		http://foo.baz/permissions	http://foo.baz/permissions
1.13.2	None	M	N	B(exp)	linkingRightsStatementIdentifierValue		General Rights Statement	General Rights Statement
					Structural map - Defines hierarchy - all objects must be in same location - strMap			
METS Structure		M	N	P(imp)	ID	A unique identifier for the element with which it is associated within the METS document that would allow the element to be referenced unambiguously from another element or document via an IDREF or an XPTR.	STR1	STR1
METS Structure		M	N	B(exp)	DO Unique Identifier	Unique identifier of Digital Object	x.pdf	a (dummy file)
METS Structure		M	N	B(exp)	DO MD Unique Identifier	Unique identifier of Metadata	x-MD.xml	a-MD.xml
METS Structure		O	R	B(exp)	Related DO ID	Related Digital Object ID - defined by business system at time of export.	nil	b.html

METS Structure		O	R	B(exp)	Related DO MD		nil	b-MD.xml
METS Structure		O	R	B(exp)	Related DO Level		nil	1
METS Structure		O	R	B(exp)	Related DO ID		nil	c.doc
METS Structure		O	R	B(exp)	Related DO ID		nil	c-MD.xml
METS Structure		O	R	B(exp)	Related DO Level		nil	2
METS		O			Other optional elements:	TYPE, LABEL		
METS	C							
METS		O			Other optional elements:	ID, TYPE, LABEL, DMDID, ADMID, ORDER, ORDERLABEL, CONTENTIDS, xlink:label		
					Structural links - record the existence of hyperlinks between items within the structural map			
					Not used			

					Behavior metadata - used to associate executable behaviors with content in the METS object - behaviorSec			
					Not used		?	

Appendix 2. Cloud Archive Ontology in XML/OWL

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
    <!ENTITY terms "http://purl.org/dc/terms/" >
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY cloudSystem "http://example.org/cloudSystem.owl#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY premis "http://example.org/CloudArchiveOntology.owl#" >
    <!ENTITY REC-skos-reference-20090818 "http://www.w3.org/TR/2009/REC-skos-reference-20090818/#" >
]>
<rdf:RDF xmlns="http://example.org/CloudArchiveOntology.owl#"
     xml:base="http://example.org/CloudArchiveOntology.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:cloudSystem="http://example.org/cloudSystem.owl#"
     xmlns:REC-skos-reference-20090818="http://www.w3.org/TR/2009/REC-skos-reference-20090818/#"
     xmlns:terms="http://purl.org/dc/terms/"
```

```
 xmlns:owl="http://www.w3.org/2002/07/owl#"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:premis="http://example.org/CloudArchiveOntology.owl#">
<owl:Ontology rdf:about="http://example.org/CloudArchiveOntology.owl">
    <terms:modified rdf:datatype="&xsd;dateTime">2012-09-01</terms:modified>
    <owl:versionInfo rdf:datatype="&xsd;string">version 0.99</owl:versionInfo>
    <owl:versionInfo>Version 0.9 Created 11/08/2012</owl:versionInfo>
    <rdfs:comment>Only covers values Mandatory values in the PREMIS data dictionary.</rdfs:comment>
    <rdfs:comment>This is an OWL Onthology for cloud based archiving systems </rdfs:comment>
    <rdfs:isDefinedBy>http://www.loc.gov/standards/premis/v2/premis-2-1.pdf</rdfs:isDefinedBy>
</owl:Ontology>

<!--
///////////////////////////////////////////////////////////////////////////////////
//
// Annotation properties
//
///////////////////////////////////////////////////////////////////////////////////
 -->
```

```
<owl:AnnotationProperty rdf:about="&premis;origin">
    <origin rdf:resource="&cloudSystem;Agents"/>
</owl:AnnotationProperty>
<owl:AnnotationProperty rdf:about="&REC-skos-reference-20090818;mappingbroadMatch"/>
<owl:AnnotationProperty rdf:about="&REC-skos-reference-20090818;mappingnarrowMatch"/>
<owl:AnnotationProperty rdf:about="&REC-skos-reference-20090818;definition">
    <rdfs:comment>
        <rdf:Description>
            <rdf:type>
                <owl:Restriction>
                    <owl:onProperty rdf:resource="&owl;topDataProperty"/>
                    <owl:someValuesFrom rdf:resource="&xsd;string"/>
                </owl:Restriction>
            </rdf:type>
        </rdf:Description>
    </rdfs:comment>
</owl:AnnotationProperty>
<rdf:Description rdf:about="&rdfs;comment">
    <rdfs:comment>
```

```
        <rdf:Description>
            <rdf:type>
                <owl:Restriction>
                    <owl:onProperty rdf:resource="&owl;topDataProperty"/>
                    <owl:someValuesFrom rdf:resource="&xsd;string"/>
                </owl:Restriction>
            </rdf:type>
        </rdf:Description>
    </rdfs:comment>
</rdf:Description>
<owl:AnnotationProperty rdf:about="&REC-skos-reference-20090818;mappingrelatedMatch"/>
<owl:AnnotationProperty rdf:about="&premis;Layer"/>
<owl:AnnotationProperty rdf:about="&owl;cardinality"/>
<owl:AnnotationProperty rdf:about="&REC-skos-reference-20090818;inScheme">
    <REC-skos-reference-20090818:inScheme>
        <rdf:Description>
            <rdf:type>
                <owl:Restriction>
                    <owl:onProperty rdf:resource="&owl;topDataProperty"/>
                    <owl:allValuesFrom rdf:resource="&xsd;anyURI"/>
```

```
                </owl:Restriction>
            </rdf:type>
        </rdf:Description>
    </REC-skos-reference-20090818:inScheme>
</owl:AnnotationProperty>
<owl:AnnotationProperty rdf:about="&REC-skos-reference-20090818;mappingexactMatch"/>
<owl:AnnotationProperty rdf:about="&REC-skos-reference-20090818;mappingcloseMatch"/>



<!--
///////////////////////////////////////////////////////////////////////////////////////
//
// Datatypes
//
///////////////////////////////////////////////////////////////////////////////////////
 -->

<!--
///////////////////////////////////////////////////////////////////////////////////////
```

```
//
// Object Properties
//
///////////////////////////////////////////////////////////////////////////////
 -->

<!-- http://example.org/CloudArchiveOntology.owl#contentLocation -->

<owl:ObjectProperty rdf:about="&premis;contentLocation">
    <rdfs:seeAlso rdf:datatype="&xsd;string">Storage class definition and ContentLocation class definition</rdfs:seeAlso>
    <Layer>Preservation</Layer>
    <rdfs:range rdf:resource="&premis;ContentLocation"/>
    <rdfs:domain rdf:resource="&premis;PreservationStorage"/>
</owl:ObjectProperty>

<!-- http://example.org/CloudArchiveOntology.owl#createsFile -->

<owl:ObjectProperty rdf:about="&premis;createsFile">
    <rdfs:comment rdf:datatype="&xsd;string">The creation of any file saved to PreservationStorage. </rdfs:comment>
    <Layer>SaaS</Layer>
```

```
    <rdfs:domain rdf:resource="&premis;Software"/>
    <rdfs:range rdf:resource="&premis;File"/>
    <rdfs:subPropertyOf rdf:resource="&cloudSystem;createsObject"/>
</owl:ObjectProperty>

<!-- http://example.org/CloudArchiveOntology.owl#createsRegistrationRequest -->

<owl:ObjectProperty rdf:about="&premis;createsRegistrationRequest">
    <rdfs:seeAlso rdf:datatype="&xsd;string">Registration Request class definition</rdfs:seeAlso>
    <Layer>SaaS</Layer>
    <rdfs:domain rdf:resource="&premis;CreatingApplication"/>
    <rdfs:range rdf:resource="&cloudSystem;RegistrationRequest"/>
    <rdfs:subPropertyOf rdf:resource="&cloudSystem;createsObject"/>
</owl:ObjectProperty>

<!-- http://example.org/CloudArchiveOntology.owl#createsRegistrationResponse -->

<owl:ObjectProperty rdf:about="&premis;createsRegistrationResponse">
    <rdfs:seeAlso rdf:datatype="&xsd;string">Registration Response class definition</rdfs:seeAlso>
    <Layer>Preservation</Layer>
```

```
        <rdfs:domain rdf:resource="&cloudSystem;PreservationService"/>
        <rdfs:range rdf:resource="&cloudSystem;RegistrationResponse"/>
        <rdfs:subPropertyOf rdf:resource="&cloudSystem;createsObject"/>
    </owl:ObjectProperty>



    <!-- http://example.org/CloudArchiveOntology.owl#createsRepresentationMetadata -->

    <owl:ObjectProperty rdf:about="&premis;createsRepresentationMetadata">
                <rdfs:seeAlso rdf:datatype="&xsd;string">Creating Application class definition and Representation Metadata class
definition</rdfs:seeAlso>
        <Layer>SaaS</Layer>
        <rdfs:domain rdf:resource="&premis;CreatingApplication"/>
        <rdfs:range rdf:resource="&cloudSystem;RepresentationMetadata"/>
        <rdfs:subPropertyOf rdf:resource="&cloudSystem;createsObject"/>
    </owl:ObjectProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#dependency -->
```

```
<owl:ObjectProperty rdf:about="&premis;dependency">
    <rdfs:seeAlso rdf:datatype="&xsd;string">Environment class definition and Dependency class definition</rdfs:seeAlso>
    <Layer>SaaS</Layer>
    <rdfs:range rdf:resource="&premis;Dependency"/>
    <rdfs:domain rdf:resource="&premis;Environment"/>
</owl:ObjectProperty>

<!-- http://example.org/CloudArchiveOntology.owl#derivationalRelationship -->

<owl:ObjectProperty rdf:about="&premis;derivationalRelationship">
    <Layer>SaaS</Layer>
    <Layer>Preservation</Layer>
    <rdfs:subPropertyOf rdf:resource="&premis;relationship"/>
    <rdfs:range rdf:resource="&cloudSystem;Object"/>
    <rdfs:domain rdf:resource="&cloudSystem;Object"/>
</owl:ObjectProperty>



<!-- http://example.org/CloudArchiveOntology.owl#environment -->
```

```
<owl:ObjectProperty rdf:about="&premis;environment">
    <rdfs:seeAlso rdf:datatype="&xsd;string">Object class definition and Environment class definition</rdfs:seeAlso>
    <Layer>PaaS</Layer>
    <Layer>SaaS</Layer>
    <rdfs:range rdf:resource="&premis;Environment"/>
</owl:ObjectProperty>

<!-- http://example.org/CloudArchiveOntology.owl#eventOutcomeDetail -->

<owl:ObjectProperty rdf:about="&premis;eventOutcomeDetail">
    <rdfs:seeAlso rdf:datatype="&xsd;string">Event class definition and EventOutcomeDetail class definition</rdfs:seeAlso>
    <Layer>Preservation</Layer>
    <rdfs:domain rdf:resource="&premis;Event"/>
    <rdfs:range rdf:resource="&premis;EventOutcomeDetail"/>
</owl:ObjectProperty>

<!-- http://example.org/CloudArchiveOntology.owl#eventOutcomeInformation -->

<owl:ObjectProperty rdf:about="&premis;eventOutcomeInformation">
```

```
        <rdfs:seeAlso rdf:datatype="&xsd;string">Event class definition and EventOutcomeInformation class definition</rdfs:seeAlso>
        <Layer>Preservation</Layer>
        <rdfs:domain rdf:resource="&premis;Event"/>
        <rdfs:range rdf:resource="&premis;EventOutcomeInformation"/>
    </owl:ObjectProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#eventType -->


    <owl:ObjectProperty rdf:about="&premis;eventType">
          <rdfs:comment rdf:datatype="&xsd;string">Data Constraint: Value should be taken from a controlled vocabulary, i.e., SKOS vocabulary. The LOC publishes a reference vocabulary for these values at: http://id.loc.gov/vocabulary/preservationEvents. One can define its own SKOS vocabulary, but for interoperability reasons, the defined concepts should be linked to the concepts of the LOC vocabulary. The LOC vocabulary concepts are also modelled as subclasses to the Event class, catching the eventType in the class definition.</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Definition: A categorization of the nature of the event.</rdfs:comment>
           <rdfs:comment rdf:datatype="&xsd;string">Example: E77[a code used within a repository for a particular event type], Ingest</rdfs:comment>
           <rdfs:comment rdf:datatype="&xsd;string">Rationale: Categorizing events will aid the preservation repository in machine processing of event information, particularly in reporting.</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: Each repository should define its own controlled vocabulary of eventType values. A suggested starter list for consideration (see also the Glossary for more detailed definitions):
capture = the process whereby a repository actively obtains an object
compression = the process of coding data to save storage space or transmission time
```

creation = the act of creating a new object

deaccession = the process of removing an object from the inventory of a repository

decompression = the process of reversing the effects of compression

decryption = the process of converting encrypted data to plaintext

deletion = the process of removing an object from repository storage

digital signature validation = the process of determining that a decrypted digital signature matches an expected value

dissemination = the process of retrieving an object from repository storage and making it available to users

fixity check = the process of verifying that an object has not been changed in a given period

ingestion = the process of adding objects to a preservation repository

message digest calculation = the process by which a message digest (“hash”) is created

migration = a transformation of an object creating a version in a more contemporary format

normalization = a transformation of an object creating a version more conducive to preservation

replication = the process of creating a copy of an object that is, bit-wise, identical to the original

validation = the process of comparing an object with a standard and noting compliance or exceptions

virus check = the process of scanning a file for malicious programs

Note that migration, normalization, and replication are more precise subtypes of the creation event. “Creation” can be used when more precise terms do not apply, for example, when a Digital Object was first created by scanning from paper.

In general, the level of specificity in recording the type of event (e.g., whether the eventType indicates a transformation, a migration or a particular method of migration) is implementation specific and will depend upon how reporting and processing is done. Recommended practice is to record detailed information about the event itself in eventDetail rather than using a very granular value for eventType.</rdfs:comment>

<Layer>Preservation</Layer>

```
    <rdfs:domain rdf:resource="&premis;Event"/>
</owl:ObjectProperty>

<!-- http://example.org/CloudArchiveOntology.owl#fixity -->

<owl:ObjectProperty rdf:about="&premis;fixity">
    <rdfs:seeAlso rdf:datatype="&xsd;string">ObjectCharacteristics class definition and Fixity class definition</rdfs:seeAlso>
    <Layer>Preservation</Layer>
    <rdfs:range rdf:resource="&premis;Fixity"/>
    <rdfs:domain rdf:resource="&premis;ObjectCharacteristics"/>
</owl:ObjectProperty>

<!-- http://example.org/CloudArchiveOntology.owl#formatDesignation -->

<owl:ObjectProperty rdf:about="&premis;formatDesignation">
    <rdfs:seeAlso rdf:datatype="&xsd;string">Format class definition and FormatDesignation class definition</rdfs:seeAlso>
    <Layer>PaaS</Layer>
    <rdfs:domain rdf:resource="&premis;Format"/>
    <rdfs:range rdf:resource="&premis;FormatDesignation"/>
</owl:ObjectProperty>
```

```
    <!-- http://example.org/CloudArchiveOntology.owl#formatRegistry -->

    <owl:ObjectProperty rdf:about="&premis;formatRegistry">
        <rdfs:seeAlso rdf:datatype="&xsd;string">Format class definition and FormatRegistry class definition</rdfs:seeAlso>
        <Layer>PaaS</Layer>
        <rdfs:range rdf:resource="&premis;FormatRegistry"/>
        <rdfs:domain rdf:resource="&premis;Format"/>
    </owl:ObjectProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#identifier -->

    <owl:ObjectProperty rdf:about="&premis;identifier">
        <rdfs:seeAlso rdf:datatype="&xsd;string">Agent class definition and AgentIdentifier class definition</rdfs:seeAlso>
        <rdfs:seeAlso rdf:datatype="&xsd;string">Dependency class definition and DependencyIdentifier class definition</rdfs:seeAlso>
        <rdfs:seeAlso rdf:datatype="&xsd;string">Event class definition and EventIdentifier class definition</rdfs:seeAlso>
                    <rdfs:seeAlso rdf:datatype="&xsd;string">LicenseInformation class definition and LicenseIdentifier class
definition</rdfs:seeAlso>
        <rdfs:seeAlso rdf:datatype="&xsd;string">Object class definition and ObjectIdentifier class definition</rdfs:seeAlso>
                  <rdfs:seeAlso rdf:datatype="&xsd;string">RightsStatement class definition and RightsStatementIdentifier class
definition</rdfs:seeAlso>
```

```
    <Layer>All</Layer>
    <rdfs:range rdf:resource="&premis;Identifier"/>
    <rdfs:domain rdf:resource="&owl;Thing"/>
</owl:ObjectProperty>

<!-- http://example.org/CloudArchiveOntology.owl#inhibitors -->

<owl:ObjectProperty rdf:about="&premis;inhibitors">
    <rdfs:seeAlso rdf:datatype="&xsd;string">ObjectCharacteristics class definition and Inhibitors class definition</rdfs:seeAlso>
    <Layer>Preservation</Layer>
    <rdfs:range rdf:resource="&premis;Inhibitors"/>
    <rdfs:domain rdf:resource="&cloudSystem;Object"/>
</owl:ObjectProperty>

<!-- http://example.org/CloudArchiveOntology.owl#keyInformation -->

<owl:ObjectProperty rdf:about="&premis;keyInformation">
    <rdfs:seeAlso rdf:datatype="&xsd;string">Signature class definition and KeyInformation class definition</rdfs:seeAlso>
    <Layer>SaaS</Layer>
    <rdfs:domain rdf:resource="&premis;Signature"/>
```

```
    </owl:ObjectProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#linkingAgent -->


    <owl:ObjectProperty rdf:about="&premis;linkingAgent">
        <rdfs:seeAlso rdf:datatype="&xsd;string">Agent class definition</rdfs:seeAlso>
        <rdfs:comment rdf:datatype="&xsd;string">Definition: link to the associated Agent.</rdfs:comment>
          <rdfs:comment>Rationale: Digital provenance requiers often that relationships between agents and events are documented. The
role of the associated agent may need to be documented. For this, a SKOS vocabulary can be used. The LOC will publish a vocabulary at
http://id.loc.gov/, denoting the agent&apos;s role. These vocabulary will publish the concepts also as subproperties to the
linkingAgent property, for denoting the role of the agent in the event or rightsstatement.</rdfs:comment>
        <Layer>Preservation</Layer>
        <rdfs:domain rdf:resource="&premis;Event"/>
        <rdfs:range rdf:resource="&cloudSystem;Agents"/>
    </owl:ObjectProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#linkingEvent -->


    <owl:ObjectProperty rdf:about="&premis;linkingEvent">
        <rdfs:comment rdf:datatype="&xsd;string">Definition: The event associated with the object.</rdfs:comment>
        <rdfs:seeAlso rdf:datatype="&xsd;string">Object class definition and Event class definition</rdfs:seeAlso>
```

```
        <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: Use to link to events that are not associated with relationships between objects, such as format validation, virus checking, etc.</rdfs:comment>
        <Layer>Preservation</Layer>
        <rdfs:range rdf:resource="&premis;Event"/>
        <rdfs:domain rdf:resource="&cloudSystem;Agents"/>
    </owl:ObjectProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#linkingIntellectualEntity -->


    <owl:ObjectProperty rdf:about="&premis;linkingIntellectualEntity">
        <rdfs:comment rdf:datatype="&xsd;string">Definition: An intellectual entity associated with the object.</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: Use to link to an intellectual entity that is related to the object. This may be a link to descriptive metadata that describes the intellectual entity or some other surrogate for it that can be referenced. This link will likely be to an identifier of an object that is at a higher conceptual level than the object for which the metadata is provided, for example, to a collection or parent object.</rdfs:comment>
        <Layer>SaaS</Layer>
        <rdfs:range rdf:resource="&premis;IntellectualEntity"/>
        <rdfs:domain rdf:resource="&cloudSystem;Object"/>
    </owl:ObjectProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#linkingObject -->
```

```
    <owl:ObjectProperty rdf:about="&premis;linkingObject">
                <rdfs:comment rdf:datatype="&xsd;string">Definition: Information about an object associated with an event or rightsstatement.</rdfs:comment>
        <rdfs:seeAlso rdf:datatype="&xsd;string">Event and RightsStatement class definition and Object class definition</rdfs:seeAlso>
          <rdfs:comment rdf:datatype="&xsd;string">Rationale: Digital provenance often requires that relationships between objects and events are documented. / Rights statements must be associated with the objects to which they pertain, either by linking from the rights statement to the object(s) or by linking from the object(s) to the rights statement. This provides the mechanism for the link from the rights statement to an object. For denoting the role of the object, when related to an event, the ontology has two subproperties of linkingObject, i.e., linkingSourceObject and linkingOutcomeObject, for specifying the role of the object in the event.</rdfs:comment>
        <rdfs:domain rdf:resource="&premis;Event"/>
        <rdfs:domain rdf:resource="&premis;RightsStatement"/>
        <rdfs:range rdf:resource="&cloudSystem;Object"/>
    </owl:ObjectProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#linkingOutcomeObject -->


    <owl:ObjectProperty rdf:about="&premis;linkingOutcomeObject">
        <rdfs:seeAlso rdf:datatype="&xsd;string">Linking Object object property</rdfs:seeAlso>
        <Layer>Preservation</Layer>
        <rdfs:subPropertyOf rdf:resource="&premis;linkingObject"/>
        <rdfs:range rdf:resource="&cloudSystem;Object"/>
    </owl:ObjectProperty>
```

```
    <!-- http://example.org/CloudArchiveOntology.owl#linkingRightsStatement -->


    <owl:ObjectProperty rdf:about="&premis;linkingRightsStatement">
        <rdfs:comment rdf:datatype="&xsd;string">Definition: A rights statement associated with the object.</rdfs:comment>
          <rdfs:comment rdf:datatype="&xsd;string">Rationale: A repository may choose to link from a rights statement to an object or
from an object to a rights statement or both.</rdfs:comment>
        <rdfs:seeAlso rdf:datatype="&xsd;string">RightsStatement class definition</rdfs:seeAlso>
        <Layer>Preservation</Layer>
        <Layer>SaaS</Layer>
        <rdfs:range rdf:resource="&premis;RightsStatement"/>
        <rdfs:domain rdf:resource="&cloudSystem;Object"/>
    </owl:ObjectProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#linkingSourceObject -->


    <owl:ObjectProperty rdf:about="&premis;linkingSourceObject">
        <rdfs:seeAlso rdf:datatype="&xsd;string">Linking Object object property</rdfs:seeAlso>
        <Layer>Preservation</Layer>
        <rdfs:subPropertyOf rdf:resource="&premis;linkingObject"/>
```

```
        <rdfs:range rdf:resource="&cloudSystem;Object"/>
    </owl:ObjectProperty>



    <!-- http://example.org/CloudArchiveOntology.owl#messageDigestAlgorithm -->


    <owl:ObjectProperty rdf:about="&premis;messageDigestAlgorithm">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
           <rdfs:comment rdf:datatype="&xsd;string">Data Constraint: Value should be taken from a controlled vocabulary, i.e., SKOS
vocabulary. The LOC publishes a reference vocabulary for these values at: http://id.loc.gov/vocabulary/cryptographicHashFunctions. One
can define its own SKOS vocabulary, but for interoperability reasons, the defined concepts should be linked to the concepts of the LOC
vocabulary.</rdfs:comment>
           <rdfs:comment rdf:datatype="&xsd;string">Definition: The specific algorithm used to construct the message digest for the
Digital Object.</rdfs:comment>
              <rdfs:comment rdf:datatype="&xsd;string">Examples: MD5, Adler-32, HAVAL, SHA-1, SHA-256, SHA-384, SHA-512, TIGER,
WHIRLPOOL</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">The specific algorithm used to construct the message digest for the
Digital Object</rdfs:comment>
        <rdfs:comment>Mandatory</rdfs:comment>
        <rdfs:comment>Not Repeatable</rdfs:comment>
```

```
        <origin>Preservation Service</origin>
        <Layer>Preservation</Layer>
        <rdfs:domain rdf:resource="&premis;Fixity"/>
        <rdfs:domain rdf:resource="&cloudSystem;Object"/>
    </owl:ObjectProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#objectCharacteristics -->


    <owl:ObjectProperty rdf:about="&premis;objectCharacteristics">
          <rdfs:comment rdf:datatype="&xsd;string"> Technical properties of a file or bitstream that are applicable to all or most
formats</rdfs:comment>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <rdfs:seeAlso rdf:datatype="&xsd;string">Object class definition and ObjectCharacteristics class definition</rdfs:seeAlso>
        <rdfs:comment>Not repeatable</rdfs:comment>
        <rdfs:comment>Mandatory</rdfs:comment>
        <origin>Preservation Service</origin>
        <Layer>Preservation</Layer>
        <rdfs:range rdf:resource="&premis;ObjectCharacteristics"/>
        <rdfs:domain rdf:resource="&cloudSystem;Object"/>
    </owl:ObjectProperty>
```

```
    <!-- http://example.org/CloudArchiveOntology.owl#platform -->


    <owl:ObjectProperty rdf:about="&premis;platform">

            <rdfs:comment rdf:datatype="&xsd;string">Creation / Maintenance Notes: In a cloud environment, hardware information is difficult to provide and subject to change without notice. In this ontology, the concept of Hardware has be replaced by Platform. Platform covers cloud storage.</rdfs:comment>

          <rdfs:comment rdf:datatype="&xsd;string">Definition: Platform components needed by the software referenced in swName or the human user of the referenced software.</rdfs:comment>

        <rdfs:seeAlso rdf:datatype="&xsd;string">Environment class definition</rdfs:seeAlso>

        <owl:deprecated rdf:datatype="&xsd;string">Hardware</owl:deprecated>

        <rdfs:range rdf:resource="&premis;Platform"/>

    </owl:ObjectProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#preservationLevelRole -->


    <owl:ObjectProperty rdf:about="&premis;preservationLevelRole">

           <rdfs:comment rdf:datatype="&xsd;string">Data Constraint: Value should be taken from a controlled vocabulary, i.e., SKOS vocabulary. The LOC publishes a reference vocabulary for these values at: http://id.loc.gov/vocabulary/preservationLevelRole. One can define its own SKOS vocabulary, but for interoperability reasons, the defined concepts should be linked to the concepts of the LOC vocabulary.</rdfs:comment>

         <rdfs:comment rdf:datatype="&xsd;string">Definition: A value indicating the context in which a set of preservation options is applicable.</rdfs:comment>
```

<rdfs:comment rdf:datatype="&xsd;string">Examples: requirement, intention, capability</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Rationale: Repositories may assign preservationLevelValues in different contexts which must be differentiated, and may need to record more than one context.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Usage Notes: This optional semantic unit qualifies the sense or context in which the preservationLevelValue in the current preservationLevel container is applied.

For example, a repository may have a legislated obligation to “fully preserve” object X (which is of format F) but is presently only capable of preserving objects of format F at a “bit-level”. The repository may need to record both the required or intended level of preservation (e.g. preservationLevelRole=“requirement”) and the current capability (e.g. preservationLevelRole=“capability”).

In transferring custody of material from one repository to another, it may also be important for the receiving repository to know the sense in which preservationLevelValue should be understood. A receiving repository may not need to know a “capability” preservation level of which the transferring repository was capable (as this will have little bearing on its own capabilities), but it needs to know any preservation level “requirements” for material for which it is now taking responsibility.

It is good practice to specify preservationLevelRole for clarity even if the repository only assigns preservationLevelValue in one sense or context. If more than one preservationLevel is recorded, preservationLevelRole should always be supplied.

If more than one sense or context needs to be expressed for the same object, (e.g. both the “requirement” and “capability” are recorded), separate preservationLevel containers should be used.</rdfs:comment>

<Layer>Preservation</Layer>

<rdfs:range rdf:resource="&premis;PreservationLevel"/>

<rdfs:domain rdf:resource="&cloudSystem;PreservationService"/>

</owl:ObjectProperty>

<!-- http://example.org/CloudArchiveOntology.owl#preservationServiceObjectRights -->

```
    <owl:ObjectProperty rdf:about="&premis;preservationServiceObjectRights">
        <Layer rdf:datatype="&xsd;string">Rights granted to the Preservation Service to access objects in Preservation Storage.</Layer>
        <Layer>Preservation</Layer>
        <rdfs:range rdf:resource="&cloudSystem;ObjectRights"/>
        <rdfs:domain rdf:resource="&cloudSystem;Object"/>
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#preservationServicePreservationStorageRights -->


    <owl:ObjectProperty rdf:about="&premis;preservationServicePreservationStorageRights">
          <rdfs:comment rdf:datatype="&xsd;string">Access credentials for the PreservationService to access and perform operations in
Preservation Storage</rdfs:comment>
        <Layer>Preservation</Layer>
        <rdfs:range rdf:resource="&premis;PreservationStorage"/>
        <rdfs:domain rdf:resource="&cloudSystem;PreservationService"/>
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>
```

```
    <!-- http://example.org/CloudArchiveOntology.owl#relatedObject -->


    <owl:ObjectProperty rdf:about="&premis;relatedObject">
        <Layer>SaaS</Layer>
        <rdfs:range rdf:resource="&premis;RelatedObjectIdentification"/>
        <rdfs:domain rdf:resource="&cloudSystem;Object"/>
    </owl:ObjectProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#relationship -->


    <owl:ObjectProperty rdf:about="&premis;relationship">
        <Layer>SaaS</Layer>
          <rdfs:comment>Usage Notes: Many formats for representing structural information may be used instead of the semantic units specified here. This information must be known, and some implementations may know it by using other structures.

Structural relationships at the file level are necessary to reconstruct a representation in order to ascertain that the representation is

renderable.

A record of structural relationships at the representation level may be necessary to render the representation. Structural relationships at the bitstream level can relate bitstreams within a file. Derivative relationships at the file and representation level are

important for documenting digital provenance.</rdfs:comment>
```

```
        <rdfs:comment>Definition: This property links one object to one or more other objects.</rdfs:comment>

            <rdfs:comment>The LOC will provide a SKOS vocabulary, where the concepts can also be used as object properties at http://id.loc.gov/. These relationships will capture the relationship type and subtype. One can define its own relationships, but for interoperability reasons, these should be linked to the LOC vocabulary.</rdfs:comment>

        <Layer>Preservation</Layer>

         <rdfs:comment>Rationale: A preservation repository must know how to assemble complex objects from component parts (structural relationships) and rigorously track digital provenance (derivation relationships).</rdfs:comment>

    </owl:ObjectProperty>




    <!-- http://example.org/CloudArchiveOntology.owl#rightsGranted -->


    <owl:ObjectProperty rdf:about="&premis;rightsGranted">

        <rdfs:seeAlso rdf:datatype="&xsd;string">RightsStatement class definition and RightsGranted class definition</rdfs:seeAlso>

        <Layer>SaaS</Layer>

        <rdfs:range rdf:resource="&premis;RightsGranted"/>

        <rdfs:domain rdf:resource="&premis;RightsStatement"/>

    </owl:ObjectProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#rightsStatement -->
```

```
    <owl:ObjectProperty rdf:about="&premis;rightsStatement">
        <rdfs:seeAlso rdf:datatype="&xsd;string">Rights class definition and RightsStatement class definition</rdfs:seeAlso>
        <Layer>SaaS</Layer>
                            <REC-skos-reference-20090818:mappingbroadMatch>http://purl.org/dc/elements/1.1/rights</REC-skos-reference-
20090818:mappingbroadMatch>
        <rdfs:domain rdf:resource="&premis;Rights"/>
        <rdfs:range rdf:resource="&premis;RightsStatement"/>
    </owl:ObjectProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#sendsBitstream -->

    <owl:ObjectProperty rdf:about="&premis;sendsBitstream">
        <Layer>PaaS</Layer>
        <rdfs:range rdf:resource="&premis;Bitstream"/>
        <rdfs:subPropertyOf rdf:resource="&cloudSystem;sendsObject"/>
    </owl:ObjectProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#sendsRepresentationMetadata -->
```

```
<owl:ObjectProperty rdf:about="&premis;sendsRepresentationMetadata">
    <Layer>SaaS</Layer>
    <rdfs:domain rdf:resource="&premis;CreatingApplication"/>
    <rdfs:range rdf:resource="&cloudSystem;RepresentationMetadata"/>
    <rdfs:subPropertyOf rdf:resource="&cloudSystem;sendsObject"/>
</owl:ObjectProperty>

<!-- http://example.org/CloudArchiveOntology.owl#signature -->

<owl:ObjectProperty rdf:about="&premis;signature">
    <rdfs:seeAlso rdf:datatype="&xsd;string">SignatureInformation class definition and Signature class definition</rdfs:seeAlso>
    <Layer>SaaS</Layer>
    <rdfs:range rdf:resource="&premis;SignatureInformation"/>
</owl:ObjectProperty>

<!-- http://example.org/CloudArchiveOntology.owl#signatureInformation -->

<owl:ObjectProperty rdf:about="&premis;signatureInformation">
    <rdfs:seeAlso rdf:datatype="&xsd;string">Object class definition and SignatureInformation class definition</rdfs:seeAlso>
    <Layer>Preservation</Layer>
```

```
        <rdfs:range rdf:resource="&premis;SignatureInformation"/>
    </owl:ObjectProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#signatureMethod -->


    <owl:ObjectProperty rdf:about="&premis;signatureMethod">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <rdfs:comment rdf:datatype="&xsd;string">A designation for the encryption and hash algorithms used for
signature generation</rdfs:comment>
          <rdfs:comment rdf:datatype="&xsd;string">Data Constraint: Value should be taken from a controlled vocabulary, i.e., SKOS vocabulary. The LOC publishes a reference vocabulary for these values at: http://id.loc.gov/vocabulary/cryptographicHashFunctions. One can define its own SKOS vocabulary, but for interoperability reasons, the defined concepts should be linked to the concepts of the LOC vocabulary.</rdfs:comment>
         <rdfs:comment rdf:datatype="&xsd;string">Definition: A designation for the encryption and hash algorithms used for signature generation.</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Examples: DSA-SHA1, RSA-SHA1</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Rationale: The same algorithms must be used for signature validation.</rdfs:comment>
           <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: Recommended practice is to encode the encryption algorithm first, followed by a hyphen, followed by the hash (message digest) algorithm.</rdfs:comment>
        <origin>Creating Application (Registration)</origin>
        <Layer>Preservation</Layer>
```

```
        <rdfs:comment>Not repeatable</rdfs:comment>
        <rdfs:comment>Mandatory</rdfs:comment>
        <rdfs:domain rdf:resource="&premis;Signature"/>
    </owl:ObjectProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#significantProperties -->

    <owl:ObjectProperty rdf:about="&premis;significantProperties">
        <rdfs:seeAlso rdf:datatype="&xsd;string">Object class definition and SignificantProperties class definition</rdfs:seeAlso>
                          <REC-skos-reference-20090818:mappingrelatedMatch>http://purl.org/dc/elements/1.1/format</REC-skos-reference-
20090818:mappingrelatedMatch>
        <Layer>Preservation</Layer>
                        <REC-skos-reference-20090818:mappingrelatedMatch>http://purl.org/dc/elements/1.1/coverage</REC-skos-reference-
20090818:mappingrelatedMatch>
        <rdfs:range rdf:resource="&premis;SignificantProperties"/>
    </owl:ObjectProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#software -->

    <owl:ObjectProperty rdf:about="&premis;software">
        <rdfs:seeAlso rdf:datatype="&xsd;string">Environment class definition and Software class definition</rdfs:seeAlso>
```

```
    <Layer>SaaS</Layer>
    <rdfs:range rdf:resource="&premis;Software"/>
</owl:ObjectProperty>

<!-- http://example.org/CloudArchiveOntology.owl#storage -->

<owl:ObjectProperty rdf:about="&premis;storage">
    <rdfs:seeAlso rdf:datatype="&xsd;string">Object class definition and Storage class definition</rdfs:seeAlso>
    <rdfs:comment>PaaS</rdfs:comment>
    <rdfs:range rdf:resource="&premis;PreservationStorage"/>
    <rdfs:domain rdf:resource="&premis;Bitstream"/>
    <rdfs:domain rdf:resource="&premis;File"/>
    <rdfs:domain rdf:resource="&premis;Representation"/>
    <rdfs:domain rdf:resource="&cloudSystem;InformationPackage"/>
</owl:ObjectProperty>

<!-- http://example.org/CloudArchiveOntology.owl#structuralRelationship -->

<owl:ObjectProperty rdf:about="&premis;structuralRelationship">
    <Layer>Preservation</Layer>
```

```
    <Layer>SaaS</Layer>
    <rdfs:subPropertyOf rdf:resource="&premis;relationship"/>
    <rdfs:domain rdf:resource="&cloudSystem;Object"/>
    <rdfs:range rdf:resource="&cloudSystem;Object"/>
</owl:ObjectProperty>

<!-- http://example.org/CloudArchiveOntology.owl#termOfGrant -->

<owl:ObjectProperty rdf:about="&premis;termOfGrant">
    <rdfs:seeAlso rdf:datatype="&xsd;string">RightsGranted class definition and TermOfGrant class definition</rdfs:seeAlso>
    <rdfs:comment>SaaS</rdfs:comment>
    <rdfs:domain rdf:resource="&premis;RightsGranted"/>
    <rdfs:range rdf:resource="&premis;TermOfGrant"/>
</owl:ObjectProperty>

<!-- http://example.org/cloudSystem.owl#converts -->

<owl:ObjectProperty rdf:about="&cloudSystem;converts">
    <Layer>Preservation</Layer>
    <rdfs:seeAlso>Preservation Service Class and Conversion Class definition</rdfs:seeAlso>
```

```
        <rdfs:range rdf:resource="&premis;Representation"/>
        <rdfs:domain rdf:resource="&cloudSystem;PreservationService"/>
        <rdfs:range rdf:resource="&cloudSystem;RepresentationMetadata"/>
    </owl:ObjectProperty>

    <!-- http://example.org/cloudSystem.owl#createsObject -->

    <owl:ObjectProperty rdf:about="&cloudSystem;createsObject">
        <rdfs:domain rdf:resource="&cloudSystem;Agents"/>
        <rdfs:range rdf:resource="&cloudSystem;Object"/>
    </owl:ObjectProperty>

    <!-- http://example.org/cloudSystem.owl#createsPackage -->

    <owl:ObjectProperty rdf:about="&cloudSystem;createsPackage">
                <rdfs:seeAlso rdf:datatype="&xsd;string">SPreservation Service class definition and Information Package class
definition</rdfs:seeAlso>
        <Layer>Preservation</Layer>
        <rdfs:domain rdf:resource="&cloudSystem;PreservationService"/>
        <rdfs:range rdf:resource="&cloudSystem;InformationPackage"/>
```

```
        <rdfs:subPropertyOf rdf:resource="&cloudSystem;createsObject"/>
    </owl:ObjectProperty>

    <!-- http://example.org/cloudSystem.owl#createsRepresentation -->

    <owl:ObjectProperty rdf:about="&cloudSystem;createsRepresentation">
                    <rdfs:seeAlso rdf:datatype="&xsd;string">Creating Application class definition and Representation class definition</rdfs:seeAlso>
        <Layer>SaaS</Layer>
        <rdfs:domain rdf:resource="&premis;CreatingApplication"/>
        <rdfs:range rdf:resource="&premis;Representation"/>
        <rdfs:subPropertyOf rdf:resource="&cloudSystem;createsObject"/>
    </owl:ObjectProperty>

    <!-- http://example.org/cloudSystem.owl#describesRepresentation -->

    <owl:ObjectProperty rdf:about="&cloudSystem;describesRepresentation">
                  <rdfs:seeAlso rdf:datatype="&xsd;string">Representation Metadata class definition and Representation class definition</rdfs:seeAlso>
        <Layer>SaaS</Layer>
        <rdfs:range rdf:resource="&premis;Representation"/>
```

```
    <rdfs:domain rdf:resource="&cloudSystem;RepresentationMetadata"/>
</owl:ObjectProperty>

<!-- http://example.org/cloudSystem.owl#registration -->

<owl:ObjectProperty rdf:about="&cloudSystem;registration">
    <rdfs:comment>Type:Event</rdfs:comment>
    <rdfs:domain rdf:resource="&cloudSystem;PreservationService"/>
    <rdfs:range rdf:resource="&cloudSystem;RegistrationRequest"/>
    <rdfs:range rdf:resource="&cloudSystem;RegistrationResponse"/>
</owl:ObjectProperty>

<!-- http://example.org/cloudSystem.owl#sendsFile -->

<owl:ObjectProperty rdf:about="&cloudSystem;sendsFile">
    <Layer>SaaS</Layer>
    <rdfs:domain rdf:resource="&premis;CreatingApplication"/>
    <rdfs:range rdf:resource="&premis;File"/>
    <rdfs:subPropertyOf rdf:resource="&cloudSystem;sendsObject"/>
</owl:ObjectProperty>
```

```
<!-- http://example.org/cloudSystem.owl#sendsObject -->

<owl:ObjectProperty rdf:about="&cloudSystem;sendsObject">
    <rdfs:domain rdf:resource="&cloudSystem;Agents"/>
    <rdfs:range>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&cloudSystem;sendsObject"/>
            <owl:someValuesFrom rdf:resource="&cloudSystem;Object"/>
        </owl:Restriction>
    </rdfs:range>
</owl:ObjectProperty>

<!-- http://example.org/cloudSystem.owl#sendsPackage -->

<owl:ObjectProperty rdf:about="&cloudSystem;sendsPackage">
    <Layer>Preservation</Layer>
    <rdfs:domain rdf:resource="&cloudSystem;PreservationService"/>
    <rdfs:range rdf:resource="&cloudSystem;InformationPackage"/>
    <rdfs:subPropertyOf rdf:resource="&cloudSystem;sendsObject"/>
```

```
</owl:ObjectProperty>

<!-- http://example.org/cloudSystem.owl#sendsRegistrationRequestTo -->

<owl:ObjectProperty rdf:about="&cloudSystem;sendsRegistrationRequestTo">
    <rdfs:seeAlso rdf:datatype="&xsd;string">Registration Request class definition</rdfs:seeAlso>
    <Layer>SaaS</Layer>
    <rdfs:domain rdf:resource="&premis;CreatingApplication"/>
    <rdfs:range rdf:resource="&cloudSystem;PreservationService"/>
    <rdfs:subPropertyOf rdf:resource="&cloudSystem;registration"/>
</owl:ObjectProperty>

<!-- http://example.org/cloudSystem.owl#sendsRegistrationResponseTo -->

<owl:ObjectProperty rdf:about="&cloudSystem;sendsRegistrationResponseTo">
    <rdfs:seeAlso rdf:datatype="&xsd;string">Registration Response class definition</rdfs:seeAlso>
    <Layer>PaaS</Layer>
    <rdfs:range rdf:resource="&premis;CreatingApplication"/>
    <rdfs:domain rdf:resource="&cloudSystem;PreservationService"/>
    <rdfs:subPropertyOf rdf:resource="&cloudSystem;registration"/>
```

```
    <owl:inverseOf rdf:resource="&cloudSystem;sendsRegistrationRequestTo"/>
</owl:ObjectProperty>

<!-- http://example.org/cloudSystem.owl#sendsRepresentation -->

<owl:ObjectProperty rdf:about="&cloudSystem;sendsRepresentation">
    <Layer>SaaS</Layer>
    <rdfs:domain rdf:resource="&premis;CreatingApplication"/>
    <rdfs:range rdf:resource="&premis;Representation"/>
    <rdfs:subPropertyOf rdf:resource="&cloudSystem;sendsObject"/>
</owl:ObjectProperty>

<!--
//////////////////////////////////////////////////////////////////////////////////
//
// Data properties
//
//////////////////////////////////////////////////////////////////////////////////
 -->
```

```
    <!-- http://example.org/CloudArchiveOntology.owl#act -->

    <owl:DatatypeProperty rdf:about="&premis;act">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <rdfs:comment rdf:datatype="&xsd;string">Data Constraint: Value should be taken from a controlled vocabulary.</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Definition: The action the preservation repository is allowed to take.</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: Suggested values:
replicate = make an exact copy
migrate = make a copy identical in content in a different file format
modify = make a version different in content
use = read without copying or modifying (e.g., to validate a file or run a program)
disseminate = create a copy or version for use outside of the preservation repository
delete = remove from the repository
It is up to the preservation repository to decide how granular the controlled vocabulary should be. It may be useful to employ the same
controlled values that the repository uses for eventType.</rdfs:comment>
        <rdfs:comment>Mandatory</rdfs:comment>
        <origin>Creating Application (registration)</origin>
        <rdfs:comment>Non repeatable</rdfs:comment>
        <origin>Creating Application (export)</origin>
```

```
    <rdfs:domain rdf:resource="&premis;RightsGranted"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<!-- http://example.org/CloudArchiveOntology.owl#agentIdentifierType -->

<owl:DatatypeProperty rdf:about="&premis;agentIdentifierType">
    <rdfs:comment rdf:datatype="&xsd;string"> A designation of the domain in which the agent identifier is unique</rdfs:comment>
    <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    <origin>Preservation Service</origin>
    <rdfs:comment>Mandatory</rdfs:comment>
    <rdfs:comment>Non repeatable</rdfs:comment>
    <rdfs:domain rdf:resource="&cloudSystem;Agents"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<!-- http://example.org/CloudArchiveOntology.owl#agentIdentifierValue -->

<owl:DatatypeProperty rdf:about="&premis;agentIdentifierValue">
    <rdfs:comment rdf:datatype="&xsd;string"> The value of the agentIdentifier</rdfs:comment>
```

```
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>

          <REC-skos-reference-20090818:mappingrelatedMatch rdf:datatype="&xsd;string">http://purl.org/dc/elements/1.1/contributor</REC-skos-reference-20090818:mappingrelatedMatch>

          <REC-skos-reference-20090818:mappingnarrowMatch rdf:datatype="&xsd;string">http://purl.org/dc/elements/1.1/creator</REC-skos-reference-20090818:mappingnarrowMatch>

        <origin>Preservation Service</origin>

        <rdfs:comment>Non repeatable</rdfs:comment>

        <rdfs:comment>Mandatory</rdfs:comment>

        <rdfs:domain rdf:resource="&cloudSystem;Agents"/>

        <rdfs:range rdf:resource="&xsd;anyURI"/>

    </owl:DatatypeProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#compositionLevel -->


    <owl:DatatypeProperty rdf:about="&premis;compositionLevel">

        <rdf:type rdf:resource="&owl;FunctionalProperty"/>

        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>

          <rdfs:comment rdf:datatype="&xsd;string">Creation / Maintenance Notes: Composition level will generally be supplied by the repository, which should attempt to supply this value automatically. If the object was created by the repository, the creating routine knows the composition level and can supply this metadata. If the object is being ingested by the repository, repository programs will have to attempt to identify the composition level from the object itself or from externally supplied metadata.</rdfs:comment>

        <rdfs:comment rdf:datatype="&xsd;string">Data Constraints: Non-negative integers.</rdfs:comment>
```

<rdfs:comment rdf:datatype="&xsd;string">Definition: An indication of whether the object is subject to one or more processes of decoding or unbundling.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Examples: 0, 1, 2</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Rationale: A file or bitstream can be encoded with compression, encryption, etc., or bundled with other files or bitstreams into larger packages. Knowing the order in which these actions are taken is important if the original object or objects must be recovered.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Usage Notes: A file or bitstream can be subject to multiple encodings that must be decoded in reverse order (highest to lowest). For example, file A may be compressed to create file B, which is encrypted to create file C. To recreate a copy of the base file A, one would have to unencrypt file C to create file B and then uncompress file B to create file A. A compositionLevel of zero indicates that the object is a base object and not subject to further decoding, while a level of 1 or higher indicates that one or more decodings must be applied.

Numbering goes lowest to highest (first encoded = 0). 0 is base object; 1-n are subsequent encodings.

Use 0 as the default if there is only one compositionLevel.

When multiple file objects are bundled together as filestreams within a package file object (e.g., a ZIP file), the individual filestream objects are not composition levels of the package file object. They should be considered separate objects, each with their own composition levels. For example, two encrypted files zipped together and stored in an archive as one file object would be described as three separate objects, each with its own associated metadata. The storage location of the two inner objects would point to the ZIP file, but the ZIP file itself would have only a single composition level (of zero) whose format would be “zip.”</rdfs:comment>

<rdfs:comment>Mandatory</rdfs:comment>

<origin>Preservation Service</origin>

<rdfs:comment>Not repeatable</rdfs:comment>

<rdfs:domain rdf:resource="&premis;ObjectCharacteristics"/>

<rdfs:range rdf:resource="&xsd;int"/>

</owl:DatatypeProperty>

```
    <!-- http://example.org/CloudArchiveOntology.owl#contentLocationType -->

    <owl:DatatypeProperty rdf:about="&premis;contentLocationType">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <rdfs:comment rdf:datatype="&xsd;string">Data Constraint: Value should be taken from a controlled vocabulary.</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Definition: The means of referencing the location of the content.</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Examples: URI, hdl, NTFS, EXT3, byte offset (bitstream)</rdfs:comment>
           <rdfs:comment rdf:datatype="&xsd;string">Rationale: To understand the meaning of the value it is necessary to know what
location scheme is used.</rdfs:comment>
        <origin>Storage Controller</origin>
        <rdfs:comment>Not repeatable</rdfs:comment>
        <rdfs:comment>Mandatory</rdfs:comment>
        <rdfs:domain rdf:resource="&premis;ContentLocation"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#contentLocationValue -->
```

```
    <owl:DatatypeProperty rdf:about="&premis;contentLocationValue">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
            <rdfs:comment rdf:datatype="&xsd;string">Definition: The reference to the location of the content used by the storage system.</rdfs:comment>
             <rdfs:comment rdf:datatype="&xsd;string">Examples: http://wwasearch.loc.gov/107th/200212107035/http://house.gov/langevin/ (file), c:\apache2\htdocs\index.html (file), 64 [offset from start of file c:\apache2\htdocs\image\logo.gif] (bitstream)</rdfs:comment>
          <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: This could be a fully qualified path and filename, or the information used by a resolution system (e.g., a handle) or the native information used by a storage management system. For a bitstream or filestream, this would probably be the reference point and offset of the starting position of the bitstream. It is up to the repository to determine the level of granularity that should be recorded.</rdfs:comment>
        <rdfs:comment>Not repeatable</rdfs:comment>
        <rdfs:comment>Mandatory</rdfs:comment>
        <origin>Storage Controller</origin>
        <rdfs:domain rdf:resource="&premis;ContentLocation"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#copyrightJurisdiction -->


    <owl:DatatypeProperty rdf:about="&premis;copyrightJurisdiction">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
```

```
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <rdfs:comment rdf:datatype="&xsd;string">Data Constraint: Values should be taken from ISO 3166.</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Definition: The country whose copyright laws apply.</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Examples: us, de, be</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Rationale: Copyright law can vary from country to country.</rdfs:comment>
        <rdfs:comment>Non repeatable</rdfs:comment>
        <rdfs:comment>Mandatory</rdfs:comment>
        <origin>Business System (Registration)</origin>
        <rdfs:domain rdf:resource="&premis;CopyrightInformation"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#creatingApplicationMetadataSchema -->

    <owl:DatatypeProperty rdf:about="&premis;creatingApplicationMetadataSchema">
        <Layer>SaaS</Layer>
        <REC-skos-reference-20090818:inScheme>Registration Request</REC-skos-reference-20090818:inScheme>
        <origin>Creating Application (Registration)</origin>
                    <rdfs:comment>Definition: URI of the Metadata Schema used by the Creating Application. For example,
http://dublincore.org/documents/dces/.</rdfs:comment>
```

```
        <rdfs:domain rdf:resource="&premis;CreatingApplication"/>
        <rdfs:range rdf:resource="&xsd;anyURI"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#creatingApplicationObjectTypes -->

    <owl:DatatypeProperty rdf:about="&premis;creatingApplicationObjectTypes">
        <REC-skos-reference-20090818:inScheme>Registration Request</REC-skos-reference-20090818:inScheme>
        <Layer>SaaS</Layer>
        <origin>Creating Application (Registration)</origin>
          <rdfs:comment>Definition: Formal description of object types created by the Creating Application. For example, the PRONOM Persistent Unique Identifier (PUID).</rdfs:comment>
        <rdfs:domain rdf:resource="&premis;CreatingApplication"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#creatingApplicationOwner -->

    <owl:DatatypeProperty rdf:about="&premis;creatingApplicationOwner">
        <origin>Creating Application (Registration)</origin>
```

```
        <rdfs:comment>Definition: Organisation with overall ownership and responsibility for the Creating Application. Should include
name, contact details and short description.</rdfs:comment>
        <Layer>SaaS</Layer>
        <REC-skos-reference-20090818:inScheme>Registration Request</REC-skos-reference-20090818:inScheme>
        <rdfs:domain rdf:resource="&premis;CreatingApplication"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#creatingApplicationProtocol -->

    <owl:DatatypeProperty rdf:about="&premis;creatingApplicationProtocol">
        <origin>Creating Application (Registration)</origin>
        <REC-skos-reference-20090818:inScheme>Registration Request</REC-skos-reference-20090818:inScheme>
        <rdfs:comment>Protocol supported by the Creating Application when exporting Digital Objects</rdfs:comment>
        <Layer>SaaS</Layer>
        <rdfs:domain rdf:resource="&premis;CreatingApplication"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#creatingApplicationURI -->
```

```
    <owl:DatatypeProperty rdf:about="&premis;creatingApplicationURI">
        <rdfs:comment>The URI of the Creating Application</rdfs:comment>
        <origin>Creating Application (Registration)</origin>
        <REC-skos-reference-20090818:inScheme>Registration Request</REC-skos-reference-20090818:inScheme>
        <Layer>SaaS</Layer>
        <rdfs:domain rdf:resource="&premis;CreatingApplication"/>
        <rdfs:range rdf:resource="&xsd;anyURI"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#dependencyIdentifierType -->

    <owl:DatatypeProperty rdf:about="&premis;dependencyIdentifierType">
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <rdfs:comment>Not Repeatable</rdfs:comment>
        <origin>Preservation System</origin>
        <rdfs:comment>Mandatory</rdfs:comment>
        <rdfs:comment>A designation of the domain in which the identifier of the dependent
resource is unique</rdfs:comment>
        <rdfs:domain rdf:resource="&premis;Representation"/>
```

```
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<!-- http://example.org/CloudArchiveOntology.owl#dependencyIdentifierValue -->

<owl:DatatypeProperty rdf:about="&premis;dependencyIdentifierValue">
    <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    <rdfs:comment>Mandatory</rdfs:comment>
    <origin>Creating Application (Export)</origin>
    <rdfs:comment>The value of the dependencyIdentifier</rdfs:comment>
    <rdfs:comment>Not Repeatable</rdfs:comment>
    <rdfs:domain rdf:resource="&premis;Representation"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<!-- http://example.org/CloudArchiveOntology.owl#eventDateTime -->

<owl:DatatypeProperty rdf:about="&premis;eventDateTime">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
```

<rdfs:comment rdf:datatype="&xsd;string">Data Constraint: To aid machine processing, value should use a structured form: xsd:dateTime</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Definition: The single date and time, or date and time range, at or during which the event occurred.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Example: 2001-10-26T19:32:52+00:00</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Usage Notes: Recommended practice is to record the most specific time possible and to designate the time zone.</rdfs:comment>

<REC-skos-reference-20090818:mappingcloseMatch rdf:datatype="&xsd;string">http://purl.org/dc/elements/1.1/date</REC-skos-reference-20090818:mappingcloseMatch>

<origin>Preservation Service</origin>

<rdfs:comment>Mandatory</rdfs:comment>

<rdfs:comment>Non repeatable</rdfs:comment>

<rdfs:domain rdf:resource="&premis;Event"/>

<rdfs:range rdf:resource="&xsd;dateTime"/>

</owl:DatatypeProperty>

<!-- http://example.org/CloudArchiveOntology.owl#formatName -->

<owl:DatatypeProperty rdf:about="&premis;formatName">

<rdf:type rdf:resource="&owl;FunctionalProperty"/>

<owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>

<rdfs:comment rdf:datatype="&xsd;string">Data Constraint: Value should be taken from a controlled vocabulary.</rdfs:comment>

```
        <rdfs:comment rdf:datatype="&xsd;string">Definition: A designation of the format of the file or bitstream.</rdfs:comment>

            <rdfs:comment rdf:datatype="&xsd;string">Examples: Text/sgml, image/tiff/geotiff, Adobe PDF, DES, PGP, base64, unknown, LaTex</rdfs:comment>

                <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: For unidentified formats, formatName may be recorded as “unknown”.</rdfs:comment>

              <REC-skos-reference-20090818:mappingbroadMatch rdf:datatype="&xsd;string">http://purl.org/dc/elements/1.1/type</REC-skos-reference-20090818:mappingbroadMatch>

        <rdfs:comment>Mandatory</rdfs:comment>

        <origin>Preservation Service</origin>

        <rdfs:comment>Non repeatable</rdfs:comment>

        <rdfs:domain rdf:resource="&premis;FormatDesignation"/>

        <rdfs:range rdf:resource="&xsd;string"/>

    </owl:DatatypeProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#formatRegistryKey -->


    <owl:DatatypeProperty rdf:about="&premis;formatRegistryKey">

        <rdf:type rdf:resource="&owl;FunctionalProperty"/>

        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>

          <rdfs:comment rdf:datatype="&xsd;string">Definition: The unique key used to reference an entry for this format in a format registry.</rdfs:comment>

        <rdfs:comment rdf:datatype="&xsd;string">Examples: info:gdfr/fred/f/tiff, TIFF/6.0</rdfs:comment>
```

```
        <origin>Preservation Service</origin>
        <rdfs:comment>Mandatory</rdfs:comment>
        <rdfs:comment>Not repeatable</rdfs:comment>
        <rdfs:domain rdf:resource="&premis;FormatRegistry"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#formatRegistryName -->

    <owl:DatatypeProperty rdf:about="&premis;formatRegistryName">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <rdfs:comment rdf:datatype="&xsd;string">Definition: A designation identifying the referenced format registry.</rdfs:comment>
            <rdfs:comment rdf:datatype="&xsd;string">Examples: PRONOM, www.nationalarchives.gov.uk/pronom, Representation Information
Registry Repository, FRED: A format registry demonstration, release 0.07</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: This can be a formal name, internally used name, or URI.</rdfs:comment>
        <rdfs:comment>Not repeatable</rdfs:comment>
        <origin>Preservation Service</origin>
        <rdfs:comment>Mandatory</rdfs:comment>
        <rdfs:domain rdf:resource="&premis;FormatRegistry"/>
```

```
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#identifierURI -->

    <owl:DatatypeProperty rdf:about="&premis;identifierURI">
        <rdfs:comment>All individuals in the cloud archive system should be identifiable by a unique ID. If the individual is described
in XML or a similar nested language, the identifierURI can be expressed as the path to its location in the hierarchy.</rdfs:comment>
        <origin>Preservation Service</origin>
        <rdfs:range rdf:resource="&xsd;anyURI"/>
        <rdfs:domain rdf:resource="&owl;Thing"/>
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#inhibitorType -->

    <owl:DatatypeProperty rdf:about="&premis;inhibitorType">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <rdfs:comment rdf:datatype="&xsd;string">Data Constraint: Value should be taken from a controlled vocabulary.</rdfs:comment>
```

```
        <rdfs:comment rdf:datatype="&xsd;string">Definition: The inhibitor method employed.</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Examples: DES, PGP, Blowfish, Password protection</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: Common inhibitors are encryption and password protection. When encryption
is used the type of encryption should be specifically indicated, that is, record “DES”, not “encryption”.</rdfs:comment>
        <origin>Creating Application (Export)</origin>
        <rdfs:comment>Not repeatable</rdfs:comment>
        <origin>Creating Application (Registration)</origin>
        <rdfs:comment>Mandatory</rdfs:comment>
        <rdfs:domain rdf:resource="&premis;Inhibitors"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#linkingEventIdentifierType -->

    <owl:DatatypeProperty rdf:about="&premis;linkingEventIdentifierType">
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <rdfs:comment rdf:datatype="&xsd;string">The eventIdentifierType value of the related event</rdfs:comment>
        <rdfs:comment>Mandatory</rdfs:comment>
        <rdfs:comment>Not repeatable</rdfs:comment>
        <origin>Preservation Service</origin>
```

```
    <rdfs:domain rdf:resource="&premis;Representation"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<!-- http://example.org/CloudArchiveOntology.owl#linkingEventIdentifierValue -->

<owl:DatatypeProperty rdf:about="&premis;linkingEventIdentifierValue">
    <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    <rdfs:comment rdf:datatype="&xsd;string">The eventIdentifierValue value of the related event</rdfs:comment>
    <rdfs:comment>Mandatory</rdfs:comment>
    <rdfs:comment>Not repeatable</rdfs:comment>
    <origin>Preservation Service</origin>
    <rdfs:domain rdf:resource="&premis;Representation"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<!-- http://example.org/CloudArchiveOntology.owl#linkingIntellectualEntityIdentifierType -->

<owl:DatatypeProperty rdf:about="&premis;linkingIntellectualEntityIdentifierType">
    <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
```

```
        <rdfs:comment rdf:datatype="&xsd;string">A designation of the domain within which the
linkingIntellectualEntityIdentifier is unique</rdfs:comment>
        <rdfs:comment>Not repeatable</rdfs:comment>
        <origin>Preservation Service</origin>
        <rdfs:comment>Mandatory</rdfs:comment>
        <rdfs:domain rdf:resource="&premis;Representation"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#linkingIntellectualEntityIdentifierValue -->

    <owl:DatatypeProperty rdf:about="&premis;linkingIntellectualEntityIdentifierValue">
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <rdfs:comment rdf:datatype="&xsd;string">The value of the linkingIntellectualEntityIdentifier</rdfs:comment>
        <rdfs:comment>Can be used to identify Original Metadata associated with a Representation</rdfs:comment>
        <rdfs:comment>Not repeatable</rdfs:comment>
        <rdfs:comment>Mandatory</rdfs:comment>
        <origin>Preservation Service</origin>
        <rdfs:domain rdf:resource="&premis;Representation"/>
        <rdfs:range rdf:resource="&xsd;string"/>
```

```
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#linkingRightsStatementIdentifierType -->

    <owl:DatatypeProperty rdf:about="&premis;linkingRightsStatementIdentifierType">
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <rdfs:comment rdf:datatype="&xsd;string">A designation of the domain within which the
linkingRightsStatementIdentifier is unique</rdfs:comment>
        <rdfs:comment>Not repeatable</rdfs:comment>
        <origin>Creating Application (Registration)</origin>
        <rdfs:comment>Mandatory</rdfs:comment>
        <origin>Preservation Service</origin>
        <rdfs:domain rdf:resource="&premis;Representation"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#linkingRightsStatementIdentifierValue -->

    <owl:DatatypeProperty rdf:about="&premis;linkingRightsStatementIdentifierValue">
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
```

```
        <rdfs:comment rdf:datatype="&xsd;string">The value of the linkingRightsStatementIdentifier</rdfs:comment>
        <rdfs:comment>Not repeatable</rdfs:comment>
        <origin>Preservation Service</origin>
        <rdfs:comment>Mandatory</rdfs:comment>
        <origin>Creating Application (Registration)</origin>
        <rdfs:domain rdf:resource="&premis;Representation"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#messageDigest -->

    <owl:DatatypeProperty rdf:about="&premis;messageDigest">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <rdfs:comment rdf:datatype="&xsd;string">Definition: The output of the message digest algorithm.</rdfs:comment>
                                                                    <rdfs:comment          rdf:datatype="&xsd;string">Example:
7c9b35da4f2ebd436f1cf88e5a39b3a257edf4a22be3c955ac49da2e2107b67a1924419563</rdfs:comment>
            <rdfs:comment rdf:datatype="&xsd;string">Rationale: This must be stored so that it can be compared in future fixity
checks.</rdfs:comment>
        <origin>Preservation Service</origin>
        <rdfs:comment>Not repeatable</rdfs:comment>
```

```
        <rdfs:comment>Mandatory</rdfs:comment>
        <rdfs:domain rdf:resource="&premis;Fixity"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#messageDigestAlgorithm -->

    <owl:DatatypeProperty rdf:about="&premis;messageDigestAlgorithm">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
           <rdfs:comment rdf:datatype="&xsd;string">Data Constraint: Value should be taken from a controlled vocabulary, i.e., SKOS vocabulary. The LOC publishes a reference vocabulary for these values at: http://id.loc.gov/vocabulary/cryptographicHashFunctions. One can define its own SKOS vocabulary, but for interoperability reasons, the defined concepts should be linked to the concepts of the LOC vocabulary.</rdfs:comment>
           <rdfs:comment rdf:datatype="&xsd;string">Definition: The specific algorithm used to construct the message digest for the Digital Object.</rdfs:comment>
              <rdfs:comment rdf:datatype="&xsd;string">Examples: MD5, Adler-32, HAVAL, SHA-1, SHA-256, SHA-384, SHA-512, TIGER, WHIRLPOOL</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">The specific algorithm used to construct the message digest for the
Digital Object</rdfs:comment>
        <rdfs:comment>Mandatory</rdfs:comment>
        <rdfs:comment>Not Repeatable</rdfs:comment>
```

```
        <origin>Preservation Service</origin>
        <Layer>Preservation</Layer>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#metadataSchemaInformation -->

    <owl:DatatypeProperty rdf:about="&premis;metadataSchemaInformation">
            <rdfs:comment>Information about the metadata schema used by the Creating Application. For example a URI to the Schema
namespace.</rdfs:comment>
        <origin>Creating Application</origin>
        <rdfs:comment>Not Mandatory</rdfs:comment>
        <rdfs:domain rdf:resource="&cloudSystem;RepresentationMetadata"/>
        <rdfs:range rdf:resource="&xsd;anyURI"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#objectCategory -->

    <owl:DatatypeProperty rdf:about="&premis;objectCategory">
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
```

```
        <rdfs:comment>Mandatory</rdfs:comment>
        <rdfs:comment>Not repeatable</rdfs:comment>
        <origin>Preservation Service</origin>
        <rdfs:comment>The category of object to which the metadata applies.</rdfs:comment>
        <rdfs:domain rdf:resource="&premis;Representation"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#objectCharacteristics -->


    <owl:DatatypeProperty rdf:about="&premis;objectCharacteristics">
           <rdfs:comment rdf:datatype="&xsd;string"> Technical properties of a file or bitstream that are applicable to all or most
formats</rdfs:comment>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <rdfs:seeAlso rdf:datatype="&xsd;string">Object class definition and ObjectCharacteristics class definition</rdfs:seeAlso>
        <rdfs:comment>Not repeatable</rdfs:comment>
        <rdfs:comment>Mandatory</rdfs:comment>
        <origin>Preservation Service</origin>
        <Layer>Preservation</Layer>
        <rdfs:range rdf:resource="&xsd;string"/>
```

```
    </owl:DatatypeProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#objectIdentifierType -->


    <owl:DatatypeProperty rdf:about="&premis;objectIdentifierType">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <rdfs:comment rdf:datatype="&xsd;string">Data Constraint: Value should be taken from controlled vocabulary.</rdfs:comment>
              <rdfs:comment rdf:datatype="&xsd;string">Definition: A designation of the domain within which the identifier is
unique.</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Examples: DLC, DRS, hdl:4263537</rdfs:comment>
           <rdfs:comment rdf:datatype="&xsd;string">Rationale: Identifier values cannot be assumed to be unique across domains. The
combination of identifierType and identifierValue should ensure uniqueness.</rdfs:comment>
         <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: The type of the identifier may be implicit within the repository as long
it can be explicitly communicated when the item is disseminated outside of it.</rdfs:comment>
        <rdfs:comment>Mandatory</rdfs:comment>
        <origin>Preservation Service</origin>
        <rdfs:comment>Not repeatable</rdfs:comment>
        <rdfs:domain rdf:resource="&premis;Identifier"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
```

```
    <!-- http://example.org/CloudArchiveOntology.owl#objectIdentifierValue -->


    <owl:DatatypeProperty rdf:about="&premis;objectIdentifierValue">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <rdfs:comment rdf:datatype="&xsd;string">Defnition: The value of the ObjectIdentifier.</rdfs:comment>
                <rdfs:comment rdf:datatype="&xsd;string">Examples: 0000000312 (Representation), IU2440 (File), WAC1943.56 (File),
http://nrs.harvard.edu/urn-3:FHCL.Loeb:sal (File), IU2440-1 (Bitstream)</rdfs:comment>
        <origin>Preservation Service</origin>
        <rdfs:comment>Not repeatable</rdfs:comment>
        <rdfs:comment>Mandatory</rdfs:comment>
        <rdfs:domain rdf:resource="&premis;Identifier"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#originalName -->


    <owl:DatatypeProperty rdf:about="&premis;originalName">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
```

```
        <rdfs:comment rdf:datatype="&xsd;string">Creation / Maintenance Notes: This value would always be supplied to the repository by the submitter or harvesting application. How much of the file path to preserve would be up to the repository.</rdfs:comment>

         <rdfs:comment rdf:datatype="&xsd;string">Definition: The name of the object as submitted to or harvested by the repository, before any renaming by the repository.</rdfs:comment>

        <rdfs:comment rdf:datatype="&xsd;string">Example: N419.pdf</rdfs:comment>

         <rdfs:comment rdf:datatype="&xsd;string">Rationale: The name used within the preservation repository may not be known outside of the repository. A depositor might need to request a file by its original name. Also, the repository may need to reconstruct internal links for dissemination.</rdfs:comment>

            <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: This is the name of the object as designated in the Submission Information Package (SIP). The object may have other names in different contexts. When two repositories are exchanging content, it would be important for the receiving repository to know and record the name of the representation at the originating repository. In the case of representations, this may be a directory name.</rdfs:comment>

        <origin>Creating Application (Export)</origin>

                              <REC-skos-reference-20090818:mappingexactMatch>http://purl.org/dc/elements/1.1/title</REC-skos-reference-20090818:mappingexactMatch>

        <rdfs:domain rdf:resource="&premis;Representation"/>

        <rdfs:range rdf:resource="&xsd;string"/>

    </owl:DatatypeProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#packageInformation -->


    <owl:DatatypeProperty rdf:about="&premis;packageInformation">

        <origin>Preservation Service</origin>
```

```
        <origin>Not Mandatory</origin>

           <rdfs:comment>Information about the package generated by the Preservation Service, such as Preservation Service URI, XML version and timestamp. </rdfs:comment>

        <rdfs:domain rdf:resource="&cloudSystem;Object"/>

        <rdfs:range rdf:resource="&xsd;string"/>

    </owl:DatatypeProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#platformName -->


    <owl:DatatypeProperty rdf:about="&premis;platformName">

        <rdf:type rdf:resource="&owl;FunctionalProperty"/>

        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>

          <rdfs:comment rdf:datatype="&xsd;string">Definition: Name, provider and version (if applicable) of the platform used by the Creating Application.</rdfs:comment>

                <rdfs:comment rdf:datatype="&xsd;string">Examples: High-CPU Medium Instance 1.7 GB of memory, 5 EC2 Compute Units.</rdfs:comment>

           <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: Include manufacturer when this helps to identify or disambiguate the product.
Include version for firmware or other components where that information is pertinent.</rdfs:comment>

        <rdfs:comment>Mandatory</rdfs:comment>

        <rdfs:comment>Not repeatable</rdfs:comment>

        <origin>Creating Application (registration)</origin>
```

```
    <rdfs:domain rdf:resource="&premis;Platform"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<!-- http://example.org/CloudArchiveOntology.owl#platformProvider -->

<owl:DatatypeProperty rdf:about="&premis;platformProvider">
    <origin>Not Mandatory</origin>
    <origin>Preservation Service</origin>
    <rdfs:comment>Name of Service Provider responsible for providing access to Preservation Storage</rdfs:comment>
    <rdfs:domain rdf:resource="&premis;Platform"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<!-- http://example.org/CloudArchiveOntology.owl#platformType -->

<owl:DatatypeProperty rdf:about="&premis;platformType">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    <rdfs:comment rdf:datatype="&xsd;string">Data Constraint: Value should be taken from a controlled vocabulary.</rdfs:comment>
```

<rdfs:comment rdf:datatype="&xsd;string">Definition: Class or category of the platform used by Creating Application.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Usage Notes: Suggested values: provider, API, description.</rdfs:comment>

<rdfs:comment>Mandatory</rdfs:comment>

<origin>Creating Application (registration)</origin>

<rdfs:comment>Not repeatable</rdfs:comment>

<rdfs:domain rdf:resource="&premis;Platform"/>

<rdfs:range rdf:resource="&xsd;string"/>

</owl:DatatypeProperty>

<!-- http://example.org/CloudArchiveOntology.owl#preservationLevelValue -->

<owl:DatatypeProperty rdf:about="&premis;preservationLevelValue">

<rdf:type rdf:resource="&owl;FunctionalProperty"/>

<owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>

<rdfs:comment rdf:datatype="&xsd;string">Creation / Maintenance Notes: The preservation level may be assigned by the repository or requested by the depositor and submitted as metadata.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Data Constraint: Value should be taken from a controlled vocabulary.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Definition: A value indicating the set of preservation functions expected to be applied to the object.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Examples: bit-level, full, fully supported with future migrations (File), 0</rdfs:comment>

```
          <rdfs:comment rdf:datatype="&xsd;string">Rationale: Some preservation repositories will offer multiple preservation options depending on factors such as the value or uniqueness of the material, the “preservability” of the format, the amount the customer is willing to pay, etc.</rdfs:comment>

          <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: Only one preservationLevelValue may be recorded per preservationLevel container. If a further preservationLevelValue applies to the object in a different context, a separate preservationLevel container should be repeated.</rdfs:comment>

        <rdfs:comment>Not repeatable</rdfs:comment>

        <rdfs:comment>Mandatory</rdfs:comment>

        <origin>Preservation Service</origin>

        <rdfs:domain rdf:resource="&premis;PreservationLevel"/>

        <rdfs:range rdf:resource="&xsd;string"/>

    </owl:DatatypeProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#relatedEventIdentifierType -->


    <owl:DatatypeProperty rdf:about="&premis;relatedEventIdentifierType">

        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>

        <rdfs:comment>Mandatory</rdfs:comment>

        <rdfs:comment>The eventIdentifierType of the related event</rdfs:comment>

        <rdfs:comment>Not repeatable</rdfs:comment>

        <origin>Preservation Service</origin>

        <rdfs:domain rdf:resource="&premis;Event"/>
```

```
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<!-- http://example.org/CloudArchiveOntology.owl#relatedEventIdentifierValue -->

<owl:DatatypeProperty rdf:about="&premis;relatedEventIdentifierValue">
    <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    <rdfs:comment rdf:datatype="&xsd;string">The eventIdentifierValue of the related event</rdfs:comment>
    <rdfs:comment>Mandatory</rdfs:comment>
    <origin>Preservation Service</origin>
    <rdfs:comment>Not repeatable</rdfs:comment>
    <rdfs:domain rdf:resource="&premis;Event"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<!-- http://example.org/CloudArchiveOntology.owl#relatedObjectIdentifierType -->

<owl:DatatypeProperty rdf:about="&premis;relatedObjectIdentifierType">
    <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    <rdfs:comment rdf:datatype="&xsd;string">A designation of the domain within which the identifier is unique</rdfs:comment>
```

```
    <rdfs:comment>Not Repeatable</rdfs:comment>
    <origin>Preservation Service</origin>
    <rdfs:comment>Mandatory</rdfs:comment>
    <origin>Creating Application (Registration)</origin>
    <rdfs:domain rdf:resource="&cloudSystem;Object"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<!-- http://example.org/CloudArchiveOntology.owl#relatedObjectIdentifierValue -->

<owl:DatatypeProperty rdf:about="&premis;relatedObjectIdentifierValue">
    <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    <rdfs:comment rdf:datatype="&xsd;string">The value of the related object identifier</rdfs:comment>
    <origin>Creating Application (Export)</origin>
    <rdfs:comment>Not repeatable</rdfs:comment>
    <origin>Preservation Service</origin>
    <rdfs:comment>Mandatory</rdfs:comment>
    <rdfs:domain rdf:resource="&cloudSystem;Object"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
```

```
    <!-- http://example.org/CloudArchiveOntology.owl#relationshipSubType -->

    <owl:DatatypeProperty rdf:about="&premis;relationshipSubType">
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <rdfs:comment rdf:datatype="&xsd;string">A specific characterization of the nature of the relationship
documented in relationshipType</rdfs:comment>
        <rdfs:comment>Not Repeatable</rdfs:comment>
        <rdfs:comment>Mandatory</rdfs:comment>
        <origin>Creating Application (Export)</origin>
        <origin>Preservation Service</origin>
        <rdfs:domain rdf:resource="&premis;Representation"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#relationshipType -->

    <owl:DatatypeProperty rdf:about="&premis;relationshipType">
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <rdfs:comment rdf:datatype="&xsd;string">A high-level categorization of the nature of the relationship</rdfs:comment>
```

```
        <origin>Creating Application (Export)</origin>
        <origin>Preservation Service</origin>
        <rdfs:comment>Mandatory</rdfs:comment>
        <rdfs:comment>Not Repeatable</rdfs:comment>
        <rdfs:domain rdf:resource="&premis;Representation"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#signatureEncoding -->

    <owl:DatatypeProperty rdf:about="&premis;signatureEncoding">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <rdfs:comment rdf:datatype="&xsd;string">Data Constraint: Value should be taken from a controlled vocabulary.</rdfs:comment>
                    <rdfs:comment rdf:datatype="&xsd;string">Definition: The encoding used for the values of signatureValue,
keyInformation.</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Examples: Base64, Ds:CrytoBinary</rdfs:comment>
             <rdfs:comment rdf:datatype="&xsd;string">Rationale: These values cannot be interpreted correctly if the encoding is
unknown.</rdfs:comment>
        <rdfs:comment>Mandatory</rdfs:comment>
        <origin>Creating Application (Export)</origin>
```

```
        <rdfs:comment>Not Repeatable</rdfs:comment>
        <rdfs:domain rdf:resource="&premis;Signature"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#signatureMethod -->

    <owl:DatatypeProperty rdf:about="&premis;signatureMethod">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <rdfs:comment rdf:datatype="&xsd;string">A designation for the encryption and hash algorithms used for
signature generation</rdfs:comment>
          <rdfs:comment rdf:datatype="&xsd;string">Data Constraint: Value should be taken from a controlled vocabulary, i.e., SKOS vocabulary. The LOC publishes a reference vocabulary for these values at: http://id.loc.gov/vocabulary/cryptographicHashFunctions. One can define its own SKOS vocabulary, but for interoperability reasons, the defined concepts should be linked to the concepts of the LOC vocabulary.</rdfs:comment>
         <rdfs:comment rdf:datatype="&xsd;string">Definition: A designation for the encryption and hash algorithms used for signature generation.</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Examples: DSA-SHA1, RSA-SHA1</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Rationale: The same algorithms must be used for signature validation.</rdfs:comment>
           <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: Recommended practice is to encode the encryption algorithm first, followed by a hyphen, followed by the hash (message digest) algorithm.</rdfs:comment>
```

```
        <origin>Creating Application (Registration)</origin>
        <Layer>Preservation</Layer>
        <rdfs:comment>Not repeatable</rdfs:comment>
        <rdfs:comment>Mandatory</rdfs:comment>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#signatureValidationRules -->

    <owl:DatatypeProperty rdf:about="&premis;signatureValidationRules">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
            <rdfs:comment rdf:datatype="&xsd;string">Definition: The operations to be performed in order to validate the digital signature.</rdfs:comment>
          <rdfs:comment rdf:datatype="&xsd;string">Rationale: The repository should not assume that the procedure for validating any particular signature will be known many years in the future without documentation.</rdfs:comment>
         <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: This may include the canonicalization method used before calculating the message digest, if the object was normalized before signing.
This value could also be a pointer to archive documentation.</rdfs:comment>
        <rdfs:comment>Mandatory</rdfs:comment>
        <origin>Preservation Service</origin>
```

```
        <rdfs:comment>Not Repeatable</rdfs:comment>
        <rdfs:domain rdf:resource="&premis;Signature"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#signatureValue -->

    <owl:DatatypeProperty rdf:about="&premis;signatureValue">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <rdfs:comment rdf:datatype="&xsd;string">Definition: The digital signature; a value generated from the application of a private key to a message digest.</rdfs:comment>
                                                                    <rdfs:comment          rdf:datatype="&xsd;string">Example:
juS5RhJ884qoFR8flVXd/rbrSDVGn40CapgB7qeQiT+rr0NekEQ6BHhUA8dT3+BCTBUQI0dBjlml9lwzENXvS83zRECjzXbMRTUtVZiPZG2pqKPnL2YU3A9645UCjTXU+jgFumv
7k78hieAGDzNci+PQ9KRmm//icT7JaYztgt4=</rdfs:comment>
        <origin>Creating Application (Export)</origin>
        <rdfs:comment>Mandatory</rdfs:comment>
        <rdfs:comment>Not Repeatable</rdfs:comment>
        <rdfs:domain rdf:resource="&premis;Signature"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
```

```
    <!-- http://example.org/CloudArchiveOntology.owl#significantPropertiesType -->


    <owl:DatatypeProperty rdf:about="&premis;significantPropertiesType">

          <rdfs:comment rdf:datatype="&xsd;string">Definition: The aspect, facet, or attribute of an object about which significant properties are being described.</rdfs:comment>

         <rdfs:comment rdf:datatype="&xsd;string">Examples: content, structure, behavior, page count, page width, typeface, hyperlinks (representation), image count (representation), color space [for an embedded image] (bitstream)</rdfs:comment>

        <rdfs:comment rdf:datatype="&xsd;string">Not repeatable</rdfs:comment>

        <rdfs:comment rdf:datatype="&xsd;string">Optional</rdfs:comment>

          <rdfs:comment rdf:datatype="&xsd;string">Rationale: Repositories may choose to describe significant properties based on a particular aspect or attribute of an object.</rdfs:comment>

         <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: This semantic unit is optional and may be used as part of a facet:detail pair with significantPropertiesValue.</rdfs:comment>

           <REC-skos-reference-20090818:mappingbroadMatch rdf:datatype="&xsd;string">http://purl.org/dc/elements/1.1/description</REC-skos-reference-20090818:mappingbroadMatch>

          <REC-skos-reference-20090818:mappingbroadMatch rdf:datatype="&xsd;string">http://purl.org/dc/elements/1.1/subject</REC-skos-reference-20090818:mappingbroadMatch>

        <origin>Preservation Service</origin>

        <rdfs:domain rdf:resource="&premis;Representation"/>

        <rdfs:range rdf:resource="&xsd;string"/>

        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
```

```
    </owl:DatatypeProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#significantPropertiesValue -->


    <owl:DatatypeProperty rdf:about="&premis;significantPropertiesValue">

          <rdfs:comment rdf:datatype="&xsd;string">Definition: Description of the characteristics of a particular object subjectively determined to be important to maintain through preservation actions.</rdfs:comment>

           <rdfs:comment rdf:datatype="&xsd;string">Examples: [For a Web page containing animation that is not considered essential] Content only, [For detail associated with a significantPropertiesType of &amp;quot;behavior&amp;quot;] Hyperlinks traversable, [For a Word document with embedded links that are not considered essential] Content only, [For detail associated with significantPropertiesType of &amp;quot;behavior&amp;quot;] Editable, [For detail associated with a significantPropertiesType of &amp;quot;page width&amp;quot;] 210 mm, [For a PDF with an embedded graph, where the lines&amp;apos; color determines the lines&amp;apos; meaning] Color, [For detail associated with a significantPropertiesType of &amp;quot;appearance&amp;quot;] Color</rdfs:comment>

        <rdfs:comment rdf:datatype="&xsd;string">Not repeatable</rdfs:comment>

        <rdfs:comment rdf:datatype="&xsd;string">Optional</rdfs:comment>

           <rdfs:comment rdf:datatype="&xsd;string">Rationale: Repositories may choose to describe significant properties based on a particular aspect or attribute of an object.</rdfs:comment>

        <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: If facet:detail pairs are used, the content of significantPropertiesValue should describe the significant properties of object relevant to the aspect, facet, or attribute declared in the significantPropertiesType with which it is paired. If facet:detail pairs are not used, significantPropertiesValue may be used to freely describe any characteristic of an object. significantPropertiesValue is not repeatable. Multiple significant properties should be described in separate, repeated significantProperties container units.</rdfs:comment>

        <origin>Preservation Service</origin>

        <rdfs:domain rdf:resource="&premis;Representation"/>
```

```
        <rdfs:range rdf:resource="&xsd;string"/>
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
    </owl:DatatypeProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#size -->


    <owl:DatatypeProperty rdf:about="&premis;size">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:comment rdf:datatype="&xsd;string">Creation / Maintenance Notes: Automatically obtained by the repository.</rdfs:comment>
              <rdfs:comment rdf:datatype="&xsd;string">Definition: The size in bytes of the file or bitstream stored in the repository.</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Example: 2038937</rdfs:comment>
         <rdfs:comment rdf:datatype="&xsd;string">Rationale: Size is useful for ensuring the correct number of bytes from storage have been retrieved and that an application has enough room to move or process files. It might also be used when billing for storage.</rdfs:comment>
          <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: Defining this semantic unit as size in bytes makes it unnecessary to record a unit of measurement. However, for the purpose of data exchange the unit of measurement should be stated or understood by both partners.</rdfs:comment>
            <REC-skos-reference-20090818:mappingbroadMatch rdf:datatype="&xsd;string">http://purl.org/dc/elements/1.1/description</REC-skos-reference-20090818:mappingbroadMatch>
        <origin>Preservation Service</origin>
        <rdfs:domain rdf:resource="&premis;ObjectCharacteristics"/>
```

```
        <rdfs:range rdf:resource="&xsd;long"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#startDate -->

    <owl:DatatypeProperty rdf:about="&premis;startDate">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
           <rdfs:comment rdf:datatype="&xsd;string">Data Constraint: To aid machine processing, value should use a structured form:
xsd:dateTime</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Definition: The beginning date of the permission granted.</rdfs:comment>
        <origin>Creating Application (Registration)</origin>
        <rdfs:comment>Not Repeatable</rdfs:comment>
        <rdfs:comment>Mandatory</rdfs:comment>
        <origin>Creating Application (Export)</origin>
        <rdfs:domain rdf:resource="&premis;TermOfGrant"/>
        <rdfs:range rdf:resource="&xsd;dateTime"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#statuteCitation -->
```

```
    <owl:DatatypeProperty rdf:about="&premis;statuteCitation">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <rdfs:comment rdf:datatype="&xsd;string">Definition: An identifying designation for the statute.</rdfs:comment>
          <rdfs:comment rdf:datatype="&xsd;string">Examples: Legal Deposit (Jersey) Law 200, National Library of New Zealand (Te Puna Mātauranga o Aotearoa) Act 2003 no 19 part 4 s 34</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: Use standard citation form when applicable.</rdfs:comment>
        <rdfs:comment>Not Repeatable</rdfs:comment>
        <rdfs:comment>Mandatory</rdfs:comment>
        <origin>Creating Application (Registration)</origin>
        <origin>Creating Application (Export)</origin>
        <rdfs:domain rdf:resource="&premis;StatuteInformation"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#statuteJurisdiction -->


    <owl:DatatypeProperty rdf:about="&premis;statuteJurisdiction">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
```

```
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <rdfs:comment rdf:datatype="&xsd;string">Data Constraint: Values should be taken from a controlled vocabulary.</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Definition: The country or other political body enacting the statute.</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Examples: us, de, be</rdfs:comment>
           <rdfs:comment rdf:datatype="&xsd;string">Rationale: The connection between the object and the rights granted is based on
jurisdiction.</rdfs:comment>
        <rdfs:comment>Mandatory</rdfs:comment>
        <origin>Creating Application (Export)</origin>
        <origin>Creating Application (Registration)</origin>
        <rdfs:comment>Not Repeatable</rdfs:comment>
        <rdfs:domain rdf:resource="&premis;StatuteInformation"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#storageAuthorization -->

    <owl:DatatypeProperty rdf:about="&premis;storageAuthorization">
        <origin>Preservation Service</origin>
        <rdfs:comment>The access key (or keypair) needed to access Preservation Storage.</rdfs:comment>
        <origin>Mandatory</origin>
```

```
        <rdfs:domain rdf:resource="&premis;Platform"/>
        <rdfs:range rdf:resource="&xsd;long"/>
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#storagePath -->

    <owl:DatatypeProperty rdf:about="&premis;storagePath">
        <rdfs:comment>The Preservation Storage path to where files and metadata are located.</rdfs:comment>
        <origin>Preservation Service</origin>
        <origin>Not Mandatory</origin>
        <rdfs:domain rdf:resource="&premis;PreservationStorage"/>
        <rdfs:range rdf:resource="&xsd;anyURI"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#storageURI -->

    <owl:DatatypeProperty rdf:about="&premis;storageURI">
        <rdfs:comment>The URI of the persistent cloud storage for a particular Creating Application. The StorageURI points to the root
URI of the system.</rdfs:comment>
```

```
        <origin>Preservation Service</origin>
        <origin>Not Mandatory</origin>
        <rdfs:domain rdf:resource="&premis;PreservationStorage"/>
        <rdfs:range rdf:resource="&xsd;anyURI"/>
    </owl:DatatypeProperty>

    <!-- http://example.org/CloudArchiveOntology.owl#swName -->

    <owl:DatatypeProperty rdf:about="&premis;swName">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <rdfs:comment rdf:datatype="&xsd;string">Definition: Manufacturer and title of the software application.</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Examples: Adobe Photoshop, Adobe Acrobat Reader</rdfs:comment>
          <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: Include manufacturer when this helps to identify or disambiguate the
product, for example, use “Adobe Photoshop” rather than “Photoshop.”</rdfs:comment>
        <origin>Creating Application (Registration)</origin>
        <rdfs:comment>Not Repeatable</rdfs:comment>
        <origin>Creating Application (Export)</origin>
        <rdfs:comment>Mandatory</rdfs:comment>
        <rdfs:domain rdf:resource="&premis;Software"/>
```

```
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>


    <!-- http://example.org/CloudArchiveOntology.owl#swType -->


    <owl:DatatypeProperty rdf:about="&premis;swType">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <rdfs:comment rdf:datatype="&xsd;string">Data Constraint: Value should be taken from a controlled vocabulary.</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Definition: Class or category of software.</rdfs:comment>
            <rdfs:comment rdf:datatype="&xsd;string">Rationale: Several different layers of software can be required to support an object.</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: Suggested values:
renderer = application that can display/play/execute the format instance, e.g., image viewer, video player, Java virtual machine (when the format instance is a Java class file)
ancillary = required ancillary software, e.g., run time libraries, browser plug-ins, compression/decompression routines, utilities, operating system emulators, etc.
operatingSystem = software that supports application execution, process scheduling, memory management, file systems, etc.
driver = software with the primary function of communicating between hardware and the operating system or other software.</rdfs:comment>
        <rdfs:comment>Not Repeatable</rdfs:comment>
        <origin>Creating Application (Registration)</origin>
```

```
        <rdfs:comment>Mandatory</rdfs:comment>
        <rdfs:domain rdf:resource="&premis;Software"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>


    <!--
    ///////////////////////////////////////////////////////////////////////////////////
    //
    // Classes
    //
    ///////////////////////////////////////////////////////////////////////////////////
     -->


    <!-- http://example.org/CloudArchiveOntology.owl#ContentLocation -->


    <owl:Class rdf:about="&premis;ContentLocation">
        <rdfs:subClassOf rdf:resource="&premis;Platform"/>
         <rdfs:comment rdf:datatype="&xsd;string">Creation / Maintenance Notes: A preservation repository should never refer to content
that it does not control. Therefore, the PREMIS working group assumed that the repository will always assign the contentLocation,
probably by program.</rdfs:comment>
          <rdfs:comment rdf:datatype="&xsd;string">Definition: Information needed to retrieve a file from the storage system, or to
```

access a bitstream within a file.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Usage Notes: If the preservation repository uses the objectIdentifier as a handle for retrieving data, contentLocation is implicit and does not need to be recorded.</rdfs:comment>

<Layer>PaaS</Layer>

</owl:Class>

<!-- http://example.org/CloudArchiveOntology.owl#CopyrightInformation -->

<owl:Class rdf:about="&premis;CopyrightInformation">

<rdfs:subClassOf rdf:resource="&premis;RightsStatement"/>

<rdfs:comment rdf:datatype="&xsd;string">Definition: Information about the copyright status of the object(s).</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Usage Notes: When rightsBasis is “copyright”, copyrightInformation should be provided. Repositories may need to extend this with more detailed information. See the California Digital Library's copyrightMD schema (www.cdlib.org/inside/projects/rights/schema/) for an example of a more detailed scheme.</rdfs:comment>

<Layer>SaaS</Layer>

</owl:Class>

<!-- http://example.org/CloudArchiveOntology.owl#CreatingApplication -->

<owl:Class rdf:about="&premis;CreatingApplication">

<rdfs:subClassOf rdf:resource="&premis;Software"/>

<rdfs:comment rdf:datatype="&xsd;string">Creation / Maintenance Notes: If the object was created by the repository, assignment of creating application information should be straightforward.

If the object was created outside the repository, it is possible this information could be supplied by the depositor. It might also be extracted from the file itself; the name of the creating application is often embedded within the file.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Definition: Information about the application that created the object.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Rationale: Information about the creating application, including the version of the application and the date the file was created, can be useful for problem solving purposes. For example, it is not uncommon for certain versions of software to be known for causing conversion errors or introducing artifacts. It is also useful to determine which rendering software is available for the Digital Object. For example, if you know that the Distiller program created the PDF file, you know it will be renderable with (among other programs) Adobe Reader.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Usage Notes: This semantic unit applies to both objects created external to the repository and subsequently ingested, and to objects created by the repository, for example, through migration events.

The creatingApplication container is repeatable if more than one application processed the object in turn. For example, a file could be created by Microsoft Word and later turned into a PDF using Adobe Acrobat. Details of both the Word and Acrobat applications may be recorded. However, if both files are stored in the repository, each file should be completely described as an Object entity and linked by using relationship information with a relationshipType “derivation.”

It may also be repeated to record the creating application before the object was ingested as well as the creating application used as part of the ingest process. For example, an HTML file was created pre-ingest using Dreamweaver, and the Web crawler Heritrix then captured a snapshot of the files as part of the ingest.

The amount of information needed for creatingApplication given here is minimal. For more granularity, extensibility is provided.

Rather than having each repository record this locally, it would be preferable to have a registry of this information similar to format or environment registries.</rdfs:comment>

<rdfs:comment>SaaS Class</rdfs:comment>

<Layer>SaaS</Layer>

</owl:Class>

```
    <!-- http://example.org/CloudArchiveOntology.owl#FormatRegistry -->


    <owl:Class rdf:about="&premis;FormatRegistry">
        <rdfs:subClassOf rdf:resource="&premis;Software"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="&premis;formatRegistryKey"/>
                <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="&premis;formatRegistryName"/>
                <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
            </owl:Restriction>
        </rdfs:subClassOf>
         <rdfs:comment rdf:datatype="&xsd;string">Definition: Identifies and/or gives further information about the format by reference
to an entry in a format registry.</rdfs:comment>
         <rdfs:comment rdf:datatype="&xsd;string">Rationale: If central format registries are available to the preservation repository,
they may provide an excellent way of referencing detailed format information.</rdfs:comment>
```

<rdfs:comment rdf:datatype="&xsd;string">Usage Notes: Either formatDesignation or at least one instance of formatRegistry is required. If more than one formatRegistry needs to be recorded the format container should be repeated to include each additional set of formatRegistry information.

The PREMIS working group assumed that a number of format registries will be developed and maintained to support digital preservation efforts. The proposal for a Global Digital Format Registry (GDFR) (http://hul.harvard.edu/gdfr/documents.html#data), for example, would create a network-accessible registry designed to store detailed specifications on formats and profiles.</rdfs:comment>

<Layer>Preservation</Layer>

</owl:Class>

<!-- http://example.org/CloudArchiveOntology.owl#LicenseInformation -->

<owl:Class rdf:about="&premis;LicenseInformation">

<rdfs:subClassOf rdf:resource="&premis;RightsStatement"/>

<rdfs:comment rdf:datatype="&xsd;string">Definition: Information about a license or other agreement granting permissions related to an object.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Usage Note: When rightsBasis is “license”, licenseInformation should be provided.</rdfs:comment>

<Layer>SaaS</Layer>

</owl:Class>

<!-- http://example.org/CloudArchiveOntology.owl#Platform -->

```
    <owl:Class rdf:about="&premis;Platform">
        <rdfs:subClassOf rdf:resource="&premis;Environment"/>
        <Layer>Definition: Platform is a collective term to describe the technological environment in which Digital Objects are stored. It covers both general characteristics, such as database type and access information (Preservation Storage) and the information about where individual objects are stored.     </Layer>
        <Layer>Rationale: Platform information is necesary for the Preservation Service to provide continued access to Digital Objects. </Layer>
        <Layer>PaaS</Layer>
    </owl:Class>


    <!-- http://example.org/CloudArchiveOntology.owl#PreservationStorage -->


    <owl:Class rdf:about="&premis;PreservationStorage">
        <rdfs:subClassOf rdf:resource="&premis;Platform"/>
            <rdfs:comment rdf:datatype="&xsd;string">Definition: Information about how and where a file is stored in the storage system.</rdfs:comment>
         <rdfs:comment rdf:datatype="&xsd;string">Rationale: It is necessary for a repository to associate the contentLocation with the storageMedium.</rdfs:comment>
          <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: Normally there would be a single storage location and medium for an object, because an object in another location would be considered a different object. The storage composite should be repeated if there are two or more copies that are identical bit-wise and managed as a unit except for the medium on which they are stored. They must have a single objectIdentifier and be managed as a single object by the repository.
Although this semantic unit is mandatory, both of its subunits are optional. At least one subunit (i.e. either contentLocation or
```

storageMedium) must be present or both may be used.</rdfs:comment>

        <Layer>PaaS</Layer>

    </owl:Class>

    <!-- http://example.org/CloudArchiveOntology.owl#Software -->

    <owl:Class rdf:about="&premis;Software">

        <rdfs:subClassOf rdf:resource="&premis;Environment"/>

          <rdfs:comment rdf:datatype="&xsd;string">Creation / Maintenance Notes: If recording this explicitly, many different software environments may apply; for example, a particular object such as a PDF file may be viewable by several versions of several applications running under several operating systems and operating system versions. Although at least one software environment should be recorded, it is not necessary to record them all and each repository will have to make its own decisions about which software environments to record.

Also, what appears to the user as a single rendering program can have many dependencies, including system utilities, runtime libraries, and so on, which each might have their own dependencies in turn.

As with environment, metadata may be more efficiently managed in conjunction with a format registry either internal or external to a repository. In the absence of a global mechanism, repositories may be forced to develop their own local “registries” relating format to software environment.</rdfs:comment>

        <rdfs:comment rdf:datatype="&xsd;string">Definition: Software required to render or use the object.</rdfs:comment>

    </owl:Class>

    <!-- http://example.org/CloudArchiveOntology.owl#StatuteInformation -->

```
<owl:Class rdf:about="&premis;StatuteInformation">
    <rdfs:subClassOf rdf:resource="&premis;RightsStatement"/>
    <rdfs:subClassOf>
        <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
                <owl:Restriction>
                    <owl:onProperty rdf:resource="&premis;statuteCitation"/>
                    <owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
                    <owl:onDataRange rdf:resource="&xsd;string"/>
                </owl:Restriction>
                <owl:Restriction>
                    <owl:onProperty rdf:resource="&premis;statuteJurisdiction"/>
                    <owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
                    <owl:onDataRange rdf:resource="&xsd;string"/>
                </owl:Restriction>
            </owl:intersectionOf>
        </owl:Class>
    </rdfs:subClassOf>
    <rdfs:comment rdf:datatype="&xsd;string">Definition: Information about the statute allowing use of the object.</rdfs:comment>
          <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: When rightsBasis is “statute”, statuteInformation should be
```

```
provided.</rdfs:comment>
        <Layer>SaaS</Layer>
    </owl:Class>

    <!-- http://example.org/CloudArchiveOntology.owl#Bitstream -->

    <owl:Class rdf:about="&premis;Bitstream">
        <rdfs:subClassOf rdf:resource="&cloudSystem;Object"/>
        <Layer>PaaS</Layer>
              <rdfs:comment>Contiguous or non-contiguous data within a file that has meaningful properties for preservation
purposes.</rdfs:comment>
    </owl:Class>

    <!-- http://example.org/CloudArchiveOntology.owl#File -->

    <owl:Class rdf:about="&premis;File">
        <rdfs:subClassOf rdf:resource="&cloudSystem;Object"/>
          <rdfs:comment rdf:datatype="&xsd;string">Definition: A named and ordered sequence of bytes that is known to an operating
system.</rdfs:comment>
        <rdfs:seeAlso rdf:datatype="&xsd;string">Object class definition</rdfs:seeAlso>
        <Layer>SaaS</Layer>
```

```
</owl:Class>

<!-- http://example.org/CloudArchiveOntology.owl#Harvest -->

<owl:Class rdf:about="&premis;Harvest">
    <rdfs:subClassOf rdf:resource="&premis;Event"/>
</owl:Class>

<!-- http://example.org/CloudArchiveOntology.owl#Notification -->

<owl:Class rdf:about="&premis;Notification">
    <rdfs:subClassOf rdf:resource="&premis;Event"/>
</owl:Class>

<!-- http://example.org/CloudArchiveOntology.owl#ObjectCreation -->

<owl:Class rdf:about="&premis;ObjectCreation">
    <rdfs:subClassOf rdf:resource="&premis;Event"/>
</owl:Class>
```

```
    <!-- http://example.org/CloudArchiveOntology.owl#Representation -->


    <owl:Class rdf:about="&premis;Representation">
        <rdfs:subClassOf rdf:resource="&cloudSystem;Object"/>
        <Layer>SaaS</Layer>
        <rdfs:comment>A Digital Object instantiating or embodying an Intellectual Entity. A representation is the set of stored digital
files and structural metadata needed to provide a complete and reasonable rendition of the Intellectual Entity.</rdfs:comment>
    </owl:Class>


    <!-- http://example.org/CloudArchiveOntology.owl#Submission -->


    <owl:Class rdf:about="&premis;Submission">
        <rdfs:subClassOf rdf:resource="&premis;Event"/>
    </owl:Class>


    <!-- http://example.org/CloudArchiveOntology.owl#Dependency -->


    <owl:Class rdf:about="&premis;Dependency">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
          <rdfs:comment rdf:datatype="&xsd;string">Creation / Maintenance Notes: Recommended practice is for a repository to archive
objects on which other objects depend. These may be sent by the submitter of the primary object, or they may in some cases be
```

automatically obtained by the repository. For example, a markup file will often contain links to other objects it requires such as DTDs or XML Schema. If it does, these objects can often be identified by the link and downloaded by the repository.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Definition: Information about a non-software component or associated file needed in order to use or render the representation or file, for example, a schema, a DTD, or an entity file declaration.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Usage Notes: This semantic unit is for additional objects that are necessary to render a file or representation, not for required software or hardware. It may also be used for a non-executable component of the object, such as a font or style sheet. For things that the software requires, see swDependency.

This semantic unit does not include objects required by structural relationships, such as child content objects (e.g., figures that are part of an article), which are recorded under relationship with a relationshipType of “structural”.

It is up to the repository to determine what constitutes a dependency in the context of the designated community.

The objects noted may be internal or external to the preservation repository.</rdfs:comment>

<Layer>Preservation</Layer>

</owl:Class>

<!-- http://example.org/CloudArchiveOntology.owl#Environment -->

<owl:Class rdf:about="&premis;Environment">

<rdfs:subClassOf rdf:resource="&cloudSystem;Agents"/>

<rdfs:comment rdf:datatype="&xsd;string">Creation / Maintenance Notes: This information may be omitted when the repository is doing only bit-level preservation on the object.

Rather than having each repository record this locally, it would be preferable to have a registry of environment information similar to proposed registries of format information.

Repositories may choose to design mechanisms for inheritance, so that if the environment required for each file within a representation

is identical to the environment recorded for the representation as a whole, it is not necessary to store this information in each file.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Definition: Hardware/software combinations supporting use of the object.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Rationale: Environment is the means by which the user renders and interacts with content. Separation of digital content from its environmental context can result in the content becoming unusable.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Usage Notes: All of this semantic units' subunits are optional. At least one subunit (i.e. environmentNote, dependency, software, hardware, and/or environmentExtension) must be present if this container is included.</rdfs:comment>

</owl:Class>

<!-- http://example.org/CloudArchiveOntology.owl#Event -->

<owl:Class rdf:about="&premis;Event">

<rdfs:subClassOf rdf:resource="&owl;Thing"/>

<rdfs:comment rdf:datatype="&xsd;string">Entity properties:

Must be related to one or more objects.

Can be related to one or more agents.

Links between entities may be recorded from either direction and need not be bi-directional.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">The Event entity aggregates information about an action that involves one or more Object entities. Metadata about an Event would normally be recorded and stored separately from the Digital Object.

Whether or not a preservation repository records an Event depends upon the importance of the event. Actions that modify objects should always be recorded. Other actions such as copying an object for backup purposes may be recorded in system logs or an audit trail but

not necessarily in an Event entity.

Mandatory semantic units are: eventIdentifier, eventType, and eventDateTime.</rdfs:comment>

    </owl:Class>

    <!-- http://example.org/CloudArchiveOntology.owl#EventOutcomeDetail -->

    <owl:Class rdf:about="&premis;EventOutcomeDetail">

        <rdfs:subClassOf rdf:resource="&owl;Thing"/>

                <rdfs:comment rdf:datatype="&xsd;string">Definition: A detailed description of the result or product of the event.</rdfs:comment>

         <rdfs:comment rdf:datatype="&xsd;string">Rationale: An event outcome may be sufficiently complex that a coded description is not adequate to document it.</rdfs:comment>

         <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: This may be used to record all error and warning messages issued by a program involved in the event or to record a pointer to an error log.

If the event was a validity check (e.g., profile conformance) any anomalies or quirks discovered would be recorded here.

All subunits of this semantic unit are optional. At least one subunit (i.e. eventOutcomeDetailNote and/or eventOutcomeDetailExtension) must be present if this container is included.</rdfs:comment>

        <Layer>Preservation</Layer>

    </owl:Class>

    <!-- http://example.org/CloudArchiveOntology.owl#EventOutcomeInformation -->

```
    <owl:Class rdf:about="&premis;EventOutcomeInformation">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
        <rdfs:comment rdf:datatype="&xsd;string">Definition: Information about the outcome of an event.</rdfs:comment>
          <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: A repository may wish to supplement a coded eventOutcome value with additional information in eventOutcomeDetail. Since events may have more than one outcome, the container is repeatable.
All subunits of this semantic unit are optional. At least one subunit (i.e. eventOutcome or eventOutcomeDetail) must be present if this container is included.</rdfs:comment>
        <Layer>Preservation</Layer>
    </owl:Class>


    <!-- http://example.org/CloudArchiveOntology.owl#Fixity -->


    <owl:Class rdf:about="&premis;Fixity">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
              <rdfs:comment rdf:datatype="&xsd;string">Creation / Maintenance Notes: Automatically calculated and recorded by repository.</rdfs:comment>
           <rdfs:comment rdf:datatype="&xsd;string">Definition: Information used to verify whether an object has been altered in an undocumented or unauthorized way.</rdfs:comment>
         <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: To perform a fixity check, a message digest calculated at some earlier time is compared with a message digest calculated at a later time. If the digests are the same, the object was not altered in the interim. Recommended practice is to use two or more message digests calculated by different algorithms. (Note that the terms “message digest” and “checksum” are commonly used interchangeably. However, the term “checksum” is more correctly used for the product of a cyclical redundancy check (CRC), whereas the term “message digest” refers to the result of a cryptographic hash function, which is what
```

is referred to here.)

The act of performing a fixity check and the date it occurred would be recorded as an Event. The result of the check would be recorded as the eventOutcome. Therefore, only the messageDigestAlgorithm and messageDigest need to be recorded as objectCharacteristics for future comparison.

Representation level: It could be argued that if a representation consists of a single file or if all the files comprised by a representation are combined (e.g., zipped) into a single file, then a fixity check could be performed on the representation. However, in both cases the fixity check is actually being performed on a file, which in this case happens to be coincidental with a representation.

Bitstream level: Message digests can be computed for bitstreams although they are not as common as with files. For example, the JPX format, which is a JPEG2000 format, supports the inclusion of MD5 or SHA-1 message digests in internal metadata that was calculated on any range of bytes of the file.</rdfs:comment>

        <Layer>Preservation</Layer>

    </owl:Class>

    <!-- http://example.org/CloudArchiveOntology.owl#Format -->

    <owl:Class rdf:about="&premis;Format">

        <rdfs:subClassOf rdf:resource="&owl;Thing"/>

        <rdfs:subClassOf>

            <owl:Class>

                <owl:unionOf rdf:parseType="Collection">

                    <owl:Restriction>

                        <owl:onProperty rdf:resource="&premis;formatDesignation"/>

```
                    <owl:onClass rdf:resource="&premis;FormatDesignation"/>
                    <owl:maxQualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxQualifiedCardinality>
                </owl:Restriction>
                <owl:Restriction>
                    <owl:onProperty rdf:resource="&premis;formatRegistry"/>
                    <owl:onClass rdf:resource="&premis;FormatRegistry"/>
                    <owl:maxQualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxQualifiedCardinality>
                </owl:Restriction>
            </owl:unionOf>
        </owl:Class>
    </rdfs:subClassOf>
```



```
    <rdfs:comment rdf:datatype="&xsd;string">Creation / Maintenance Notes: The format of a file or bitstream should be ascertained by the repository on ingest. Even if this information is provided by the submitter, directly in metadata or indirectly via the file name extension, recommended practice is to independently identify the format by parsing the file when possible. If the format cannot be identified at the time of ingest, it is valid to record that it is unknown, but the repository should subsequently make an effort to identify the format, even if manual intervention is required.</rdfs:comment>

    <rdfs:comment rdf:datatype="&xsd;string">Definition: Identification of the format of a file or bitstream where format is the organization of digital information according to preset specifications.</rdfs:comment>

    <rdfs:comment rdf:datatype="&xsd;string">Rationale: Many preservation activities depend on detailed knowledge about the format of the Digital Object. An accurate identification of format is essential. The identification provided, whether by name or pointer into a format registry, should be sufficient to associate the object with more detailed format information.</rdfs:comment>

    <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: A bitstream embedded within a file may have different characteristics than the larger file. For example, a bitstream in LaTex format could be embedded within an SGML file, or multiple images using
```

different colorspaces could be embedded within a TIFF file. format must be recorded for every object. When the bitstream format can be recognized by the repository and the repository might want to treat the bitstream differently from the embedding file for preservation purposes, format can be recorded for embedded bitstreams.

Although this semantic unit is mandatory, both of its subunits are optional. At least one subunit (i.e. either formatDesignation or formatRegistry) must be present if this container is included or both may be used. If the subunit (formatDesignation or formatRegistry) needs to be repeated, the entire format container is repeated. This allows for association of format designation with a particular set of format registry information. For example, if the precise format cannot be determined and two format designations are recorded, each is given within a separate format container. The format container may also be repeated for multiple format registry entries.</rdfs:comment>

        <Layer>Preservation</Layer>

        <Layer>SaaS</Layer>

    </owl:Class>

    <!-- http://example.org/CloudArchiveOntology.owl#FormatDesignation -->

    <owl:Class rdf:about="&premis;FormatDesignation">

        <rdfs:subClassOf rdf:resource="&owl;Thing"/>

        <rdfs:comment rdf:datatype="&xsd;string">Definition: An identification of the format of the object.</rdfs:comment>

          <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: Either formatDesignation or at least one instance of formatRegistry is required. Both may be included.

The most specific format (or format profile) should be recorded. A repository (or formats registry) may wish to use multipart format names (e.g., “TIFF_GeoTIFF” or “WAVE_MPEG_BWF”) to achieve this specificity.</rdfs:comment>

        <Layer>Preservation</Layer>

```
    </owl:Class>


    <!-- http://example.org/CloudArchiveOntology.owl#Identifier -->


    <owl:Class rdf:about="&premis;Identifier">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
        <rdfs:comment rdf:datatype="&xsd;string">Rationale: This class goes beyond the object identifier and related identifiers in the
Premis data dictionary. In a cloud system, entities may lie outside the control of the archiving organisation, and as such must be
uniquely identifiable.  </rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">definition:  A designation used to uniquely identify the entities within the
cloud system. Unique identifiers can be applied to Agents, Events, Objects and Rights.</rdfs:comment>
        <Layer>SaaS</Layer>
        <Layer>Preservation</Layer>
        <Layer>Interaction</Layer>
        <Layer>PaaS</Layer>
    </owl:Class>


    <!-- http://example.org/CloudArchiveOntology.owl#Inhibitors -->


    <owl:Class rdf:about="&premis;Inhibitors">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
```

<rdfs:comment rdf:datatype="&xsd;string">Creation / Maintenance Notes: Inhibitors are more likely to be present on an object ingested by the repository than applied by the repository itself. It is often not possible to tell that a file has been encrypted by parsing it; the file may appear to be ASCII text. Therefore, information about inhibitors should be supplied as metadata with submitted objects when possible.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Definition: Features of the object intended to inhibit access, use, or migration.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Rationale: Format information may indicate whether a file is encrypted, but the nature of the encryption also must be recorded, as well as the access key.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Usage Notes: Some file formats allow encryption for embedded bitstreams.

Some file formats such as PDF use passwords to control access to content or specific functions. Although this is actually implemented at the bitstream level, for preservation purposes it is effectively managed at the file level; that is, passwords would not be recorded for individually addressable bitstreams.

For certain types of inhibitor keys, more granularity may be required. If the inhibitor key information is identical to key information in digital signatures, use those semantic units.</rdfs:comment>

<Layer>SaaS</Layer>

</owl:Class>

<!-- http://example.org/CloudArchiveOntology.owl#IntellectualEntity -->

<owl:Class rdf:about="&premis;IntellectualEntity">

<rdfs:subClassOf rdf:resource="&owl;Thing"/>

<rdfs:comment rdf:datatype="&xsd;string">Definition: a set of content that is considered a single intellectual unit for purposes of management and description: for example, a particular book, map, photograph, or database. An Intellectual Entity can include other Intellectual Entities; for example, a Web site can include a Web page; a Web page can include an image. An Intellectual

Entity may have one or more digital representations.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Intellectual entities are described via Descriptive metadata models. These are very domain-specific and are out of scope for PREMIS. Examples: Dublin Core, Mets, MARC</rdfs:comment>

<Layer>SaaS</Layer>

</owl:Class>

<!-- http://example.org/CloudArchiveOntology.owl#ObjectCharacteristics -->

<owl:Class rdf:about="&premis;ObjectCharacteristics">

<rdfs:subClassOf rdf:resource="&owl;Thing"/>

<rdfs:comment rdf:datatype="&xsd;string">Definition: Technical properties of a file or bitstream that are applicable to all or most formats.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Rationale: There are some important technical properties that apply to objects of any format. Detailed definition of format-specific properties is outside the scope of this Data Dictionary, although such properties may be included within objectCharacteristicsExtension.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Usage Notes: The semantic units included in objectCharacteristics should be treated as a set of information that pertains to a single object at a single compositionLevel. Object characteristics may be repeated when an object was created by applying two or more encodings, such as compression and encryption. In this case each repetition of objectCharacteristics would have an incrementally higher compositionLevel.

When encryption is applied, the objectCharacteristics block must include an inhibitors semantic unit.

A bitstream embedded within a file may have different object characteristics than the file. Where these characteristics are relevant for preservation, they should be recorded.

When a single file is equivalent to a representation, objectCharacteristics may be applied and thus associated with the representation.

In these cases, the relationship between the file comprising the representation and other associated files may be expressed using relationshipSubType.</rdfs:comment>

        <Layer>Preservation</Layer>

    </owl:Class>

    <!-- http://example.org/CloudArchiveOntology.owl#PreservationLevel -->

    <owl:Class rdf:about="&premis;PreservationLevel">

        <rdfs:subClassOf rdf:resource="&owl;Thing"/>

        <rdfs:comment rdf:datatype="&xsd;string">Creation / Maintenance Notes: The preservation level may be assigned by the repository or requested by the depositor and submitted as metadata. The repository may also choose to record additional metadata indicating the context for the assignment of the preservation level.</rdfs:comment>

        <rdfs:comment rdf:datatype="&xsd;string">Definition: Information indicating the decision or policy on the set of preservation functions to be applied to an object and the context in which the decision or policy was made.</rdfs:comment>

        <rdfs:comment rdf:datatype="&xsd;string">Rationale: Some preservation repositories will offer multiple preservation options depending on factors such as the value or uniqueness of the material, the “preservability” of the format, the amount the customer is willing to pay, etc. The context surrounding the choice of a particular preservation option for an object may also require further explanation.</rdfs:comment>

        <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: If the repository offers only a single preservation level, this value does not need to be explicitly recorded within the repository.

Application of a particular set of preservationLevel semantic units may only cover a single representation of an object: representations in other technical forms or serving other functions may have a different preservationLevel applied.

The container may be repeated if a preservation level value needs to be recorded in additional contexts (see preservationLevelRole).</rdfs:comment>

```
        <Layer>Preservation</Layer>
    </owl:Class>

    <!-- http://example.org/CloudArchiveOntology.owl#RelatedObjectIdentification -->

    <owl:Class rdf:about="&premis;RelatedObjectIdentification">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
        <rdfs:comment xml:lang="en"> Definition: The identifier and sequential context of the related resource</rdfs:comment>
        <Layer>SaaS</Layer>
           <rdfs:comment xml:lang="en">Usage Notes: The related object may or may not be held within the preservation repository.
Recommended practice is that objects reside within the repository unless there is a good reason to reference an object outside.
Internal and external references should be clear.</rdfs:comment>
    </owl:Class>

    <!-- http://example.org/CloudArchiveOntology.owl#Rights -->

    <owl:Class rdf:about="&premis;Rights">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
        <rdfs:comment rdf:datatype="&xsd;string">Entity properties:
May be related to one or more objects.
May be related to one or more agents.
```

```
Links between entities may be recorded from either direction and need not be bi-directional.</rdfs:comment>

         <rdfs:comment rdf:datatype="&xsd;string">For the purpose of the PREMIS Data Dictionary, statements of rights and permissions are taken to be constructs that can be described as the Rights entity. Rights are entitlements allowed to agents by copyright or other intellectual property law. Permissions are powers or privileges granted by agreement between a rightsholder and another party or parties.

A repository might wish to record a variety of rights information including abstract rights statements and statements of permissions that apply to external agents and to objects not held within the repository. The minimum core rights information that a preservation repository must know, however, is what rights or permissions a repository has to carry out actions related to

objects within the repository. These may be granted by copyright law, by statute, or by a license agreement with the rightsholder.

If the repository records rights information, either rightsStatement or rightsExtension must be present.</rdfs:comment>
    </owl:Class>


    <!-- http://example.org/CloudArchiveOntology.owl#RightsGranted -->


    <owl:Class rdf:about="&premis;RightsGranted">
        <rdfs:subClassOf rdf:resource="&premis;Rights"/>
        <rdfs:subClassOf>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="&premis;termOfGrant"/>
                        <owl:onClass rdf:resource="&premis;TermOfGrant"/>
```

```
                    <owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
                </owl:Restriction>
                <owl:Restriction>
                    <owl:onProperty rdf:resource="&premis;act"/>
                    <owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
                    <owl:onDataRange rdf:resource="&xsd;string"/>
                </owl:Restriction>
            </owl:intersectionOf>
        </owl:Class>
    </rdfs:subClassOf>
            <rdfs:comment rdf:datatype="&xsd;string">Definition: The action(s) that the granting agency has allowed the repository.</rdfs:comment>
    <Layer>SaaS</Layer>
</owl:Class>

<!-- http://example.org/CloudArchiveOntology.owl#RightsStatement -->

<owl:Class rdf:about="&premis;RightsStatement">
    <rdfs:subClassOf rdf:resource="&premis;Rights"/>
    <rdfs:subClassOf>
```

```
            <owl:Restriction>
                <owl:onProperty rdf:resource="&premis;identifier"/>
                <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
            </owl:Restriction>
        </rdfs:subClassOf>
           <rdfs:comment rdf:datatype="&xsd;string">Definition: Documentation of the repository&apos;s right to perform one or more acts.</rdfs:comment>
           <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: This semantic unit is optional because in some cases rights may be unknown. Institutions are encouraged to record rights information when possible.
Either rightsStatement or rightsExtension must be present if the Rights entity is included.
The rightsStatement should be repeated when the act(s) described has more than one basis, or when different acts have different bases.</rdfs:comment>
    </owl:Class>


    <!-- http://example.org/CloudArchiveOntology.owl#Signature -->


    <owl:Class rdf:about="&premis;Signature">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
         <rdfs:comment rdf:datatype="&xsd;string">Definition: Information needed to use a digital signature to authenticate the signer of an object and/or the information contained in the object.</rdfs:comment>
         <rdfs:comment rdf:datatype="&xsd;string">Rationale: A repository may have a policy of generating digital signatures for files on ingest, or may have a need to store and later validate incoming digital signatures.</rdfs:comment>
```

```
        <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: Several of the semantic components of signatureInformation are taken from the W3C’s XML-Signature Syntax and Processing; see www.w3.org/TR/2002/REC-xmldsig-core-20020212/ for more information on the structure and application of these semantic units.</rdfs:comment>

        <Layer>SaaS</Layer>

    </owl:Class>


    <!-- http://example.org/CloudArchiveOntology.owl#SignatureInformation -->


    <owl:Class rdf:about="&premis;SignatureInformation">

        <rdfs:subClassOf rdf:resource="&owl;Thing"/>

           <rdfs:comment rdf:datatype="&xsd;string">Definition: A class for PREMIS defined and externally defined digital signature information, used to authenticate the signer of an object and/or the information contained in the object.</rdfs:comment>

         <rdfs:comment rdf:datatype="&xsd;string">Rationale: A repository may have a policy of generating digital signatures for files on ingest, or may have a need to store and later validate incoming digital signatures.</rdfs:comment>

          <rdfs:comment rdf:datatype="&xsd;string">Usage Notes: Either signature or signatureInformationExtension may be used. Use of signatureInformationExtension with the schema defined in W3C’s XML-Signature Syntax and Processing (www.w3.org/TR/2002/REC-xmldsig-core-20020212/) is encouraged when applicable. See the discussion of digital signatures on page 201 for more information on use of both PREMIS-defined and externally-defined semantic units.</rdfs:comment>

        <Layer>Preservation</Layer>

    </owl:Class>


    <!-- http://example.org/CloudArchiveOntology.owl#SignificantProperties -->
```

<owl:Class rdf:about="&premis;SignificantProperties">

<rdfs:subClassOf rdf:resource="&owl;Thing"/>

<rdfs:comment rdf:datatype="&xsd;string">Creation / Maintenance Notes: Significant properties may pertain to all objects of a certain class; for example, the repository can decide that for all PDF files, only the content need be preserved. In other cases, for example, for media art, the significant properties may be unique to each individual object. Where values are unique, they must be supplied by the submitter or provided by the curatorial staff of the repository.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Definition: Characteristics of a particular object subjectively determined to be important to maintain through preservation actions.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Rationale: Objects that have the same technical properties may still differ as to the properties that should be preserved for future presentation or use.</rdfs:comment>

<rdfs:comment rdf:datatype="&xsd;string">Usage Notes: All of this semantic unit’s subunits are optional. At least one of the significantPropertiesValue and significantPropertiesExtension subunits must be present if this container is included or both may be used.

Significant properties may be objective technical characteristics subjectively considered important, or subjectively determined characteristics. For example, a PDF may contain links that are not considered important and JavaScript that is considered important. Or future migrations of a TIFF image may require optimization for line clarity or for color; the option chosen would depend upon a curatorial judgment of the significant properties of the image.

Listing significant properties implies that the repository plans to preserve these properties across time and requires them to acceptably survive preservation action; for example, to be maintained during emulation or after format migration. It also implies that the repository would note when preservation action results in modification of significant properties.

In practice, significant properties might be used as measures of preservation success, as part of quality checking the results of a preservation action or evaluating the efficacy of a preservation method. For example, if the listed significant properties are not maintained after application of a particular preservation method, it may indicate a failure of the process or that the method is not well suited to the type of material.

More experience with digital preservation is needed to determine the best ways of representing significant properties in general, and of representing modification of significant properties.

The semantic units included in the significantProperties container aim to provide a flexible structure for describing significant properties, allowing general types of aspects, facets or attributes of an object to be declared and to be paired with specific significant details about the object pertaining to that aspect, facet or attribute.

For example, some repositories may define significant properties for objects related to facets of content, appearance, structure, behavior, and context. Examples of facet:detail pairs in this case could include:

significantPropertiesType = “content”

significantPropertiesValue = “all textual content and images”

significantPropertiesType = “behavior”

significantPropertiesValue = “editable”

Other repositories may choose to describe significant properties at a more granular attribute level; for example:

significantPropertiesType = “page count”

significantPropertiesValue = “7”

significantPropertiesType = “page width”

significantPropertiesValue = “210 mm”

Each facet:detail pair should be contained in a separate, repeated significantProperties container.

Further work on determining and describing significant properties may yield more detailed schemes to facilitate general description.

Representing modification of significant properties as a result of preservation action also requires further work. One possible way involves the use of Object and Event information: Object A has significant properties volume and timing, which are recorded as significantProperties of A. In migrated version B, the timing is modified, which is noted in the eventOutcome of the migration event. Only volume is listed as a significant property of B.</rdfs:comment>

        <Layer>Preservation</Layer>

    </owl:Class>

```
<!-- http://example.org/CloudArchiveOntology.owl#TermOfGrant -->

<owl:Class rdf:about="&premis;TermOfGrant">
    <rdfs:subClassOf rdf:resource="&premis;Rights"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&premis;startDate"/>
            <owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
            <owl:onDataRange rdf:resource="&xsd;dateTime"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:comment rdf:datatype="&xsd;string">Definition: The time period for the permissions granted.</rdfs:comment>
    <rdfs:comment rdf:datatype="&xsd;string">Rationale: The permission to preserve may be time bounded.</rdfs:comment>
    <Layer>SaaS</Layer>
</owl:Class>

<!-- http://example.org/cloudSystem.owl#ArchiveSystem -->
```

```
    <owl:Class rdf:about="&cloudSystem;ArchiveSystem">
        <rdfs:subClassOf rdf:resource="&premis;Software"/>
        <rdfs:comment rdf:datatype="&xsd;string">A system intended to preserve information for access and use by a
Designated Community.</rdfs:comment>
        <Layer>Interaction</Layer>
    </owl:Class>

    <!-- http://example.org/cloudSystem.owl#PreservationService -->

    <owl:Class rdf:about="&cloudSystem;PreservationService">
        <rdfs:subClassOf rdf:resource="&premis;Software"/>
              <rdfs:comment rdf:datatype="&xsd;string">Ensures that archive systems can access Information Objects by offering:
registration,  storage allocation, conversion,  preservation metadata extension and package creation.</rdfs:comment>
        <Layer>Preservation</Layer>
    </owl:Class>

    <!-- http://example.org/cloudSystem.owl#InformationPackage -->

    <owl:Class rdf:about="&cloudSystem;InformationPackage">
```

<rdfs:subClassOf rdf:resource="&cloudSystem;Object"/>

<rdfs:comment rdf:datatype="&xsd;string">The Content Information and associated Preservation Description Information which is needed to aid in the preservation of the Content Information. The Information Package has associated Packaging Information used to delimit and identify the Content Information and Preservation Description Information.

The Package can be either a link to the package information stored in PreservationStorage, or an xml package of preservation Metadata.</rdfs:comment>

<Layer>Interaction</Layer>

</owl:Class>

<!-- http://example.org/cloudSystem.owl#InformationPackageCreation -->

<owl:Class rdf:about="&cloudSystem;InformationPackageCreation">

<rdfs:subClassOf rdf:resource="&premis;Event"/>

<rdfs:comment rdf:datatype="&xsd;string">The creation of metadata needed to create a generic Information Package that meets the requirements of the preservation service. </rdfs:comment>

<Layer>Preservation</Layer>

</owl:Class>

<!-- http://example.org/cloudSystem.owl#ObjectConversion -->

<owl:Class rdf:about="&cloudSystem;ObjectConversion">

<rdfs:subClassOf rdf:resource="&premis;Event"/>

<rdfs:comment rdf:datatype="&xsd;string">Conversion of objects from one format to another. Includes file format conversion, aggregation and decomposition.</rdfs:comment>

<Layer>Preservation</Layer>

</owl:Class>

<!-- http://example.org/cloudSystem.owl#ObjectMetadataconversion -->

<owl:Class rdf:about="&cloudSystem;ObjectMetadataconversion">

<rdfs:subClassOf rdf:resource="&premis;Event"/>

<rdfs:comment rdf:datatype="&xsd;string">Conversio of metadata from one format to another. Includes format conversion, schema conversion and crosswalking.</rdfs:comment>

<Layer>Preservation</Layer>

</owl:Class>

<!-- http://example.org/cloudSystem.owl#ObjectRights -->

<owl:Class rdf:about="&cloudSystem;ObjectRights">

<rdfs:subClassOf rdf:resource="&premis;Rights"/>

<rdfs:comment rdf:datatype="&xsd;string">Object Rights detail the access rights to individual files and bitstreams in storage. Rights for preservation, IP and distribution rights are defined elsewhere. </rdfs:comment>

<rdfs:comment>Interaction Layer Class</rdfs:comment>

```
        <rdfs:comment>Preservation Layer Class</rdfs:comment>
        <rdfs:comment>SaaS Class</rdfs:comment>
        <rdfs:comment>PaaS Class</rdfs:comment>
    </owl:Class>

    <!-- http://example.org/cloudSystem.owl#PlatformAccessRights -->

    <owl:Class rdf:about="&cloudSystem;PlatformAccessRights">
        <rdfs:subClassOf rdf:resource="&premis;Rights"/>
          <rdfs:comment rdf:datatype="&xsd;string">Platform Access Rights detail the access rights to the storage platform containing
individual files and bitstreams.</rdfs:comment>
        <rdfs:comment>PaaS</rdfs:comment>
        <Layer>SaaS</Layer>
    </owl:Class>

    <!-- http://example.org/cloudSystem.owl#Registration -->

    <owl:Class rdf:about="&cloudSystem;Registration">
        <rdfs:subClassOf rdf:resource="&premis;Event"/>
        <Layer rdf:datatype="&xsd;string">Preservation</Layer>
```

<rdfs:comment rdf:datatype="&xsd;string">Registration is a two way process that on one hand provides the Preservation Service with information used in package creation, such as information about the creating application, metadata schema registers and crosswalks. On the other hand, it provides creating applications with information about preservation service requirements and object storage. </rdfs:comment>

</owl:Class>

<!-- http://example.org/cloudSystem.owl#RegistrationRequest -->

<owl:Class rdf:about="&cloudSystem;RegistrationRequest">

<rdfs:subClassOf rdf:resource="&cloudSystem;Object"/>

<rdfs:comment rdf:datatype="&xsd;string">Registration requests are sent from creating applications to the Preservation Service. They provide the Preservation Service with information used in package creation, such as software information about the creating application, object and metadata formats and metadata schemas in use. </rdfs:comment>

<Layer>SaaS</Layer>

</owl:Class>

<!-- http://example.org/cloudSystem.owl#RegistrationResponse -->

<owl:Class rdf:about="&cloudSystem;RegistrationResponse">

<rdfs:subClassOf rdf:resource="&cloudSystem;Object"/>

<rdfs:comment rdf:datatype="&xsd;string">The registration response can provides creating applications with information about preservation service requirements and object storage. If the information in the preservation request does not meet Preservation System requirements, the registration is unsuccessful.</rdfs:comment>

```
        <Layer>Preservation</Layer>
    </owl:Class>

    <!-- http://example.org/cloudSystem.owl#RepresentationMetadata -->

    <owl:Class rdf:about="&cloudSystem;RepresentationMetadata">
        <rdfs:subClassOf rdf:resource="&cloudSystem;Object"/>
        <Layer>SaaS</Layer>
        <rdfs:comment>Metadata from the creating application associated with a representation. </rdfs:comment>
    </owl:Class>

    <!-- http://example.org/cloudSystem.owl#Agents -->

    <owl:Class rdf:about="&cloudSystem;Agents">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
        <rdfs:comment>An Agent is a person, organization, or software program associated with preservation events in
the life of an object. </rdfs:comment>
        <rdfs:comment>May hold or grant one or more rights.
May carry out, authorize, or compel one or more events.
May create or act upon one or more objects through an event or with respect to a rights statement.</rdfs:comment>
```

```
    </owl:Class>

    <!-- http://example.org/cloudSystem.owl#Object -->

    <owl:Class rdf:about="&cloudSystem;Object">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
        <rdfs:comment>Can be associated with one or more rights statements.
Can participate in one or more events.</rdfs:comment>
        <rdfs:comment>An Object, or Digital Object, is a discrete unit of information in digital form.</rdfs:comment>
    </owl:Class>

    <!--
    //////////////////////////////////////////////////////////////////////////////////
    //
    // Individuals
    //
    //////////////////////////////////////////////////////////////////////////////////
     -->

    <!-- http://example.org/Base64 -->
```

```
    <owl:NamedIndividual rdf:about="http://example.org/Base64"/>

    <!-- http://example.org/Ds:CrytoBinary -->

    <owl:NamedIndividual rdf:about="http://example.org/Ds:CrytoBinary"/>
</rdf:RDF>

<!-- Generated by the OWL API (version 3.3.1957) http://owlapi.sourceforge.net -->
```