The Consultative Committee for Space Data Systems

Report Concerning Space Data System Standards

OPEN ARCHIVAL INFORMATION SYSTEM INTEROPERABILITY FRAMEWORK (OAIS-IF) RATIONALE, SCENARIOS, AND REQUIREMENTS

DRAFT INFORMATIONAL REPORT

CCSDS 000.0-G-0

DRAFT GREEN BOOK

October 2021

AUTHORITY

Issue:	Draft Green Book, Issue 0
Date:	July 2021
Location:	Not Applicable

(WHEN THIS INFORMATIONAL REPORT IS FINALIZED, IT WILL CONTAIN THE FOLLOWING STATEMENT OF AUTHORITY:)

FOREWORD

[Foreword text specific to this document goes here. The text below is boilerplate.]

DOCUMENT CONTROL

Document	Title and Issue	Date	Status
CCSDS 000.0-G-0	OPEN ARCHIVAL INFORMATION SYSTEM INTEROPERABILITY FRAMEWORK (OAIS-IF) RATIONALE, SCENARIOS, AND REQUIREMENTS, Draft Informational Report, Issue 0	January 2021	First draft
CCSDS 000.0-G-0	OPEN ARCHIVAL INFORMATION SYSTEM INTEROPERABILITY FRAMEWORK (OAIS-IF) RATIONALE, SCENARIOS, AND REQUIREMENTS, Draft Informational Report, Issue 0	May 2021	Updated for consistency
CCSDS 000.0-G-0	OPEN ARCHIVAL INFORMATION SYSTEM INTEROPERABILITY FRAMEWORK (OAIS-IF) RATIONALE, SCENARIOS, AND REQUIREMENTS, Draft Informational Report, Issue 0	August 2021	Updated for consistency with revised concepts
CCSDS 000.0-G-0	OPEN ARCHIVAL INFORMATION SYSTEM INTEROPERABILITY FRAMEWORK (OAIS-IF) RATIONALE, SCENARIOS, AND REQUIREMENTS, Draft Informational Report, Issue 0	October 2021	Reorganised and added ideas about protocols

CONTENTS

Contents

Table of Figures

1 INTRODUCTION

THIS DOCUMENT :

- **REFERS TO, AND USES FIGURES FROM, THE UPDATED VERSION OF THE OAIS REFERENCE MODEL (currently CCSDS 650.0-P-2.1)**
- **CONTAINS LINKS TO OTHER DOCUMENTS ON THE WEB, IN ORDER TO KEEP THE INFORMATION AVAILABLE. EVENTUALLY THESE LINKS WILL BE DELETED.**

1.1 PURPOSE

The purpose of this document is to describe the rationale, with motivating scenarios, requirements and the document tree for the CCSDS and International Organization for Standardization (ISO) Open Archival Information System Interoperability Framework (OAIS-IF). It is a supplement to the overarching OAIS [1] Reference Model standard.

The purpose of the set of standards which make up OAIS-IF is to define interfaces and services which go beyond the exchange of data (as in communications standards) and instead will enable the exchange of information.

The difference may be understood as follows. Following OAIS, data, and in particular digital data, which OAIS calls a Digital Object, is simply an object composed of a set of bit sequences. Bit sequences by themselves could mean anything. On the other hand, Information is *any type of knowledge that can be exchanged. In an exchange, it is represented by data*. It is made up of the Data Object plus what OAIS refers to as Representation Information.

The aim is to improve interoperability between users and archives by transferring Information rather than just Data, and to enable the increased usability of information of all types across all domains.

1.2 SCOPE

The OAIS-IF adds capabilities for system interoperability between more general Users (including Consumers and Producers) and archives, whether or not the archives are fully OAIS Reference Model conformant.

OAIS-IF does not define what should happen within an archive or user systems except as needed for an interoperable interface. The scope is to describe how information is transferred between users and archives in such a way that the information is understandable, as far as possible, also, if required, can be accompanied by evidence about its Authenticity, and everything needed for preservation.

1.3 APPLICABILITY

OAIS-IF should be applicable to any archive and any user. For an OAIS compliant archive, OAIS-IF should enable the archive to make not only complete Archive Information Packages (AIPs) available but also all of the individual components of its AIPs separately available. A non-OAIS compliant archive may not have all the components required for an AIP available but OAIS-IF can still be used in a way limited by what the archive can provide.

We should describe what we mean by INTEROPERABILITY

Interoperability is defined in various ways. A useful one is:

- the ability of computer systems or software to exchange and make use of information.
- "interoperability between devices made by different manufacturers“ (from Oxford Languages) and
- the degree to which two products, programs, etc.. can be used together, or the quality of being able to be used together: (Cambridge dictionary)

We can distinguish between

1) Rendered objects i.e. things we normally display (documents, images) or play (videos, sounds) can only be combined in a human’s head – unless they are treated as DATA
2) Data i.e. things which are NOT normally rendered, can be combined in a computer

We will be focussing on DATA here because we are largely interested in computer processing, but recognising that human intervention is needed.

1.4 RATIONALE

The rationale and vision for OAIS-IF is that it will enable at least the following.

- A common mechanism for users (including Producers and Consumers) of OAIS Archives when accessing many diverse kinds of archives through the OAIS-IF.
- An efficient standardized way for archives to exchange information between themselves using the same standardized OAIS-IF interfaces.
- Given broad acceptance of OAIS-IF in the OAIS community, a better chance that long-term preservation will work because future generations can easily find the resources (plug-ins, etc.) that can be used to access legacy archives.
- Enhanced capabilities for cross-discipline research when many different disciplines use the same interface. OAIS-IF will enable users to access and interpret preserved data which is normally outside of their discipline and Designated Community, if the Archive or other sources make the needed Representation Information available via OAIS-IF channels.

Digitally encoded information is very important for individuals, organisations and societies. The OAIS standard tells us how to preserve that information so that it will continue to be usable in the long term by a Designated Community. However, each Archive will have different Designated Communities. Many users of such information will not be members of any Designated Community. In that case, while their ability to understand the information is not guaranteed, OAIS and OAIS-IF still support their ability to access it, with as much Representation Information as is available either from the archive itself or from other sources.

Moreover, not all archives are OAIS conformant (as defined in the OAIS Reference Model section 1.4), yet they too contain important information. While archives that are not OAIS Reference Model conformant were not the drivers of OAIS-IF development, OAIS-IF should still be able to provide significant interoperability for most of them.

Those users within one domain, such as Astronomy, tend to have common software tools and common terminology, algorithms etc., and so are likely to be able to understand and use data from various sources, and in particular from other Astronomical archives. A person outside that community will have much greater difficulty in using such data.

The ability of such other sources to provide additional Representation Information (and PDI) via OAIS-IF technology may provide such users with the possibility of greater understanding of, and confidence in, the preserved data. That, of course, depends on such other sources providing adequate RepInfo. This is envisioned to serve both users in the public, and researchers that are studying cross-domain topics.

OAIS-IF seeks to make the information from different domains more easily usable for a broader range of customers. It will be of general benefit if all can exchange, understand and use information from all domains; it will also be of benefit to archives by increasing the use of their holdings. The OAIS-IF set of documents are designed for use by archives, information creators and users.

OAIS-IF does not seek to specify how an archive preserves information, nor how information should be created and encoded, nor what software users must employ when using information. Instead OAIS-IF defines how information should be exchanged to ensure that information will be usable. In addition, OAIS-IF allows Preservation Description Information (PDI) to be exchanged, and in particular Provenance Information, with which a user can judge Authenticity.

1.5 DOCUMENT STRUCTURE

Section 2 provides a brief overview of OAIS-IF and its uses. A number of scenarios are described in section 3; in each scenario issues, avoiding repetition, are identified. Based on these scenarios, and other considerations, a number of requirements are listed in section 4, with cross-references to the scenarios, where appropriate. Further details of the set of books which define the OAIS-IF are provided in section 6 onwards.

1.6 DEFINITIONS

1.6.1 ACRONYMS AND ABBREVIATIONS

AIP	Archival Information Package
CCSDS	Consultative Committee for Space Data Systems
DEDSL	Data Entity Specification Language
DIP	Dissemination Information Package
FITS	Flexible Image Transport System
GIS	Geographic Information System
ISO	International Organization for Standardization
OAIS	Open Archival Information System
PDI	Preservation Description Information
SIP	Submission Information Package
TEI	Text Encoding Initiative
UML	Unified Modeling Language
XML	Extensible Markup Language

1.6.2 TERMINOLOGY

Digital preservation interests a range of different communities, each with a distinct vocabulary and local definitions for key terms. A glossary is included in this document, but it is important to draw attention to the usage of several key terms.

In general, key terms in this document have been adopted from the OAIS Reference Model. One of the great strengths of the OAIS Reference Model has been to provide a common terminology made up of terms 'not already overloaded with meaning so as to reduce conveying unintended meanings' (reference [1]). Because the OAIS has become a foundational document for digital preservation, the common terms are well understood and are therefore used within this document.

The OAIS Reference Model uses 'Archive' to mean the organization responsible for digital preservation. In this document, the term 'repository' or phrase 'digital repository' is used to convey the same concept in all instances except when quoting from the OAIS Reference Model (reference [1]). It is important to understand that in all instances in this document, 'repository' and 'digital repository' are used to convey digital repositories and archives that have, or contribute to, long-term preservation responsibilities and functionality. This document uses the OAIS concept of the 'Designated Community'. A repository may have a single, generalized 'Designated Community' (e.g., every citizen of a country), while other repositories may have several, distinct user communities with highly specialized needs, each requiring different functionalities or support from the repository; this document uses the term Designated Community to cover this second case also.

1.6.2.1 Glossary

Unless otherwise indicated, other definitions are taken from the OAIS Reference Model (reference [1]) and are provided here for convenience.

Access Policy: Documented statement, authorized by the repository management, that describes the approach to be taken by the repository for providing access to objects accessioned into the repository. The Access Policy may distinguish between different types of access rights, for example between system administrators, members of the Designated Community, and general users.

Archival Information Package (AIP): An Information Package, consisting of the Content Information and the associated Preservation Description Information (PDI), which is preserved within an OAIS.

Authenticity: The degree to which a person (or system) regards an object as what it is purported to be. Authenticity is judged on the basis of evidence.

Consumer: The role played by those persons, or client systems, who interact with OAIS services to find preserved information of interest and to access that information in whatever level of detail is allowed. In addition to the normally expected entities outside the OAIS, this can also include other OAISes, as well as internal OAIS persons or systems.

Data Object: Either a Physical Object or a Digital Object.

Digital Object: An object composed of a set of bit sequences.

Dissemination Information Package (DIP): An Information Package, derived from one or more AIPs, and sent by Archives to the Consumer in response to a request to the OAIS.

Fixity Information: The information which documents the mechanisms that ensure that the Content Data Object has not been altered in an undocumented manner.

Identifier: An identifier is a name that identifies (that is, labels the identity of) either a unique object or a unique class of objects, where the "object" or class may be an idea, physical countable object (or class thereof), or physical noncountable substance (or class thereof).

Information: Any type of knowledge that can be exchanged. In an exchange, it is represented by data.

NOTE – An example of Information is a string of bits (the data) accompanied by a description of how to interpret the string of bits as numbers representing temperature observations measured in degrees Celsius (the Representation Information).

Information Object: A Data Object together with its Representation Information.

Information Package: A logical container composed of optional Information Object(s). Associated with this Information Package is Packaging Information used to delimit and identify the Information Object and optional Package Description Information used to facilitate searches for the Information Object.

Other Representation Information: A type of Representation Information which cannot easily be classified as Structure Representation Information or Semantic Representation Information. It is a type of Information Object.

NOTE – For example, software, algorithms, encryption, written instructions and many other things may be needed to understand the Content Data Object in ways exemplified by the Preservation Objectives, all of which therefore would be, by definition, Representation Information, yet would not obviously be either Structure Representation Information or Semantic Representation. Information defining how the Structure Representation Information and the Semantic Representation Information relate to each other, or software needed to process a database file would also be regarded as Other Representation Information.

Packaging Information: The information that describes how the components of an Information Package are logically or physically bound together and how to identify and extract the components. It is a type of Information Object.

Physical Object: An object (such as a moon rock, bio-specimen, microscope slide) with physically observable properties that represent information that is considered suitable for being adequately documented for preservation, distribution, and independent usage.

Preservation Description Information (PDI): The information, which along with Representation Information, is necessary for adequate preservation of the Content Data Object and which can be categorized as Provenance Information, Context Information, Reference Information, Fixity Information, and Access Rights Information. It is a type of Information Object.

NOTE – Defining PDI (as well as its components: Provenance Information, Context Information, Reference Information, Fixity Information, and Access Rights Information) as relevant to the Content Data Object does not mean that those concerns are any less important for other data objects or at other levels, for example, it is important to apply reference, fixity, provenance, context and access rights to Representation Information, or to any other information the Archive is preserving. Definition of these terms as relevant to the Content Data Object is simply to ease discussion of these concepts at the Content Data Object level.

Producer: The role played by those persons or client systems that provide the information to be preserved. This can include internal or external OAIS persons or systems.

Provenance Information: The information that documents the history of the Content Data Object. This information tells the origin or source of the Content Data Object, any changes that may have taken place since it was originated, and who has had custody of it since it was originated. The Archive is responsible for creating and preserving Provenance Information from the point of Ingest; however, earlier Provenance Information should be provided by the Producer. Provenance Information adds to the evidence to support Authenticity.

Reference Information: The information that is used as an identifier for the Content Data Object. It also includes identifiers that allow outside systems to refer unambiguously to a particular Content Data Object.

NOTE – An example of Reference Information is an ISBN.

Representation Information: The information that maps a Data Object into more meaningful concepts so that the Data Object may be understood in ways exemplified by Preservation Objectives. It is a type of Information Object.

NOTE – An example of Representation Information for a bit sequence which is a FITS file might consist of the FITS standard which defines the format plus a dictionary which defines the meaning in the file of keywords which are not part of the standard. This would then allow the information in the FITS file to be used by a computer program to display the image which may be contained in the FITS file, together with the associated coordinate system so that a human can identify objects of interest, for example stars or galaxies. Alternatively, the computer program may identify such objects automatically.

Representation Information Network: The set of Representation Information that fully describes the meaning of a Data Object. Representation Information in digital forms needs additional Representation Information so its digital forms can be understood over the Long Term. It is a type of Information Object.

Semantic Representation Information: The Representation Information that further describes the meaning of the Data Object, and its parts or elements, beyond that provided by the Structure Representation Information.

NOTE – For example, Semantic Representation Information may describe the meaning of columns, and perhaps particular values seen in the columns of a spreadsheet.

Structure Representation Information: The Representation Information that imparts information about the arrangement of and the organization of the parts or elements of the Data Object.

NOTE – For example, Structure Representation Information maps bit streams to common computer types such as characters, numbers, and pixels and aggregations of those types such as character strings and arrays.

Submission Agreement: The agreement reached between an OAIS and the Producer that specifies the intended formats and content descriptions, and any other arrangements needed (such as delivery or transmission protocols), for the Data Submission Sessions. Such specifications establish the intended deliverables from the Producer and how they are represented on each delivery through physical media or in a telecommunication dialogue.

Submission Information Package (SIP): An Information Package that is delivered by the Producer to the OAIS for use in the construction or update of one or more AIPs and/or the associated Descriptive Information.

1.7 REFERENCES

The following documents are referenced in this Report. At the time of publication, the editions indicated were valid. All documents are subject to revision, and users of this Report are encouraged to investigate the possibility of applying the most recent editions of the documents indicated below. The CCSDS Secretariat maintains a register of currently valid CCSDS documents.

[A list of documents referenced in the report goes here. See CCSDS A20.0-Y-4, *CCSDS Publications Manual* (Yellow Book, Issue 4, April 2014) for reference list format.]

[1] *Reference Model for an Open Archival Information System (OAIS)*. Recommendation for Space Data System Practices, CCSDS 650.0-M-2. Blue Book. Issue 1. Washington, D.C.: CCSDS, June 2012. [Equivalent to ISO 14721:2012.], or later Available from: https://public.ccsds.org/Pubs/650x0m2.pdf . The definitions are taken from the draft update of OAIS.

For the diagrams used in this document, see the link https://www.dropbox.com/s/c3ribtzpyz6ynh9/OAIS-IF-diagrams.pptx?dl=1

2 OVERVIEW

While section 3 explores specific user scenarios, this section (2) provides the broadest overview of OAIS-IF scenarios.

2.1 CURRENT SITUATION

An example of exchanging information between entities is a client-server system, for example a web browser and a web server, where what is being transferred between the two is encoded in HTML, with perhaps MIME encoded objects such as images, using for communication the HTTP protocol via TCP/IP. The Representation Information required to deal with the HTML and a limited number of MIME types is embedded in the web browser software, as if part of what is needed to deal with HTTP. The other parts of HTTP and TCP-IP are dealt with by the operating system and internet infrastructure.

A web browser would not be able to deal with scientific data generally except to allow it to be downloaded as a collection of bits, and as such would not be usable without a great deal of effort on the part of the user, for example to identify then read documentation and download and install software. The user may not even know where to get the documentation and software.

The following diagram illustrates the current situation when users want to combine information from multiple archives. Each archive sends whatever it wishes to send and the user must install appropriate software to deal with each, as indicated by the coloured shapes in Figure 2-1.

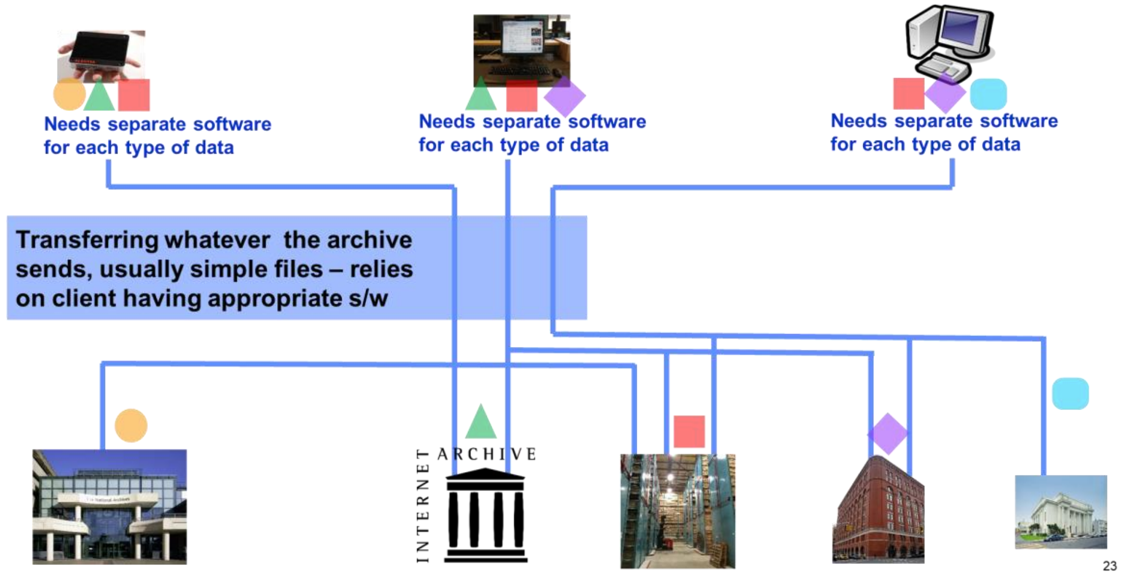
Needs separate software for each type of data
Needs separate software for each type of data
Needs separate software for each type of data
Transferring whatever the archive sends, usually simple files – relies on client having appropriate s/w
INTERNET ARCHIVE
23

Figure 2-1 Connections between clients and archives currently

In order to then combine data from many domains the user may need to go through the same process for each archive, potentially including downloading unique access software (or setting up unique configurations) for each archive. However, even then it may be difficult to combine the data without transforming each set of data to a common format - if such a suitable format exists and is usable by the users' software.

Each step outlined above is a significant hurdle to the use and integration of information from multiple sources.

2.2 WHAT OAIS-IF IS DESIGNED TO MAKE POSSIBLE

OAIS-IF defines a set of common Application Programming Interfaces (APIs) and functionalities which will allow developers to create software to provide users with usable information from an archive or set of archives. The software developers may or may not be associated with the archive, but could, for example, be supporting efforts in user groups. Using that software, consumers will be able access data and associated Representation Information (if available) from archives across disparate domains of study.

Some of the interfaces will be generally applicable, and reference implementations should be available. The implementation of other interfaces will be specific to the software being used by the archive or information creator or information user.

The following illustrates the way in which OAIS-IF will enable increased usage between two generic pieces of software, shown here as S1 and S2. The "OAIS-IF interfaces and objects" is based on the OAIS Information Model, which itself is designed to be applicable to any type of Information. The adapters allow S1 and S2 to use the OAIS-IF interfaces and objects.

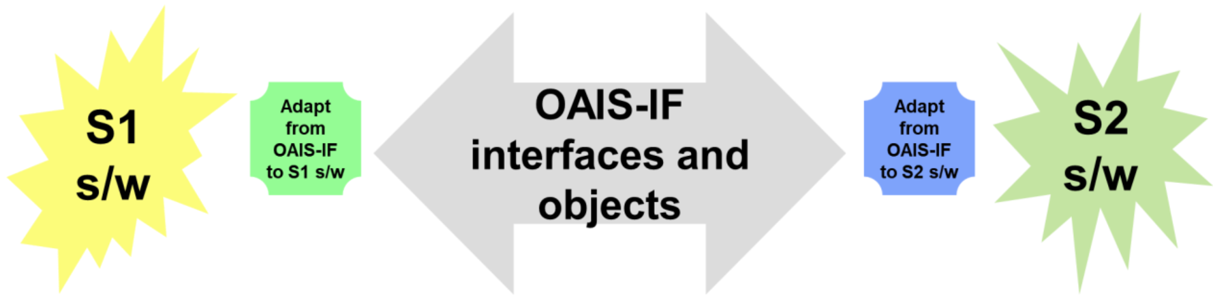
S1 s/w
Adapt from OAIS-IF to S1 s/w
OAIS-IF interfaces and objects
Adapt from OAIS-IF to S2 s/w
S2 s/w

Figure 2-2 General use of OAIS-IF

The advantage of using OAIS-related ideas is that the OAIS Information Model was designed to accommodate any type of digitally encoded Information in a way which allows that information to be usable.

This can then be generalized to show the way in which users can use their software to access and use information from multiple different archives for which their software is not originally designed.

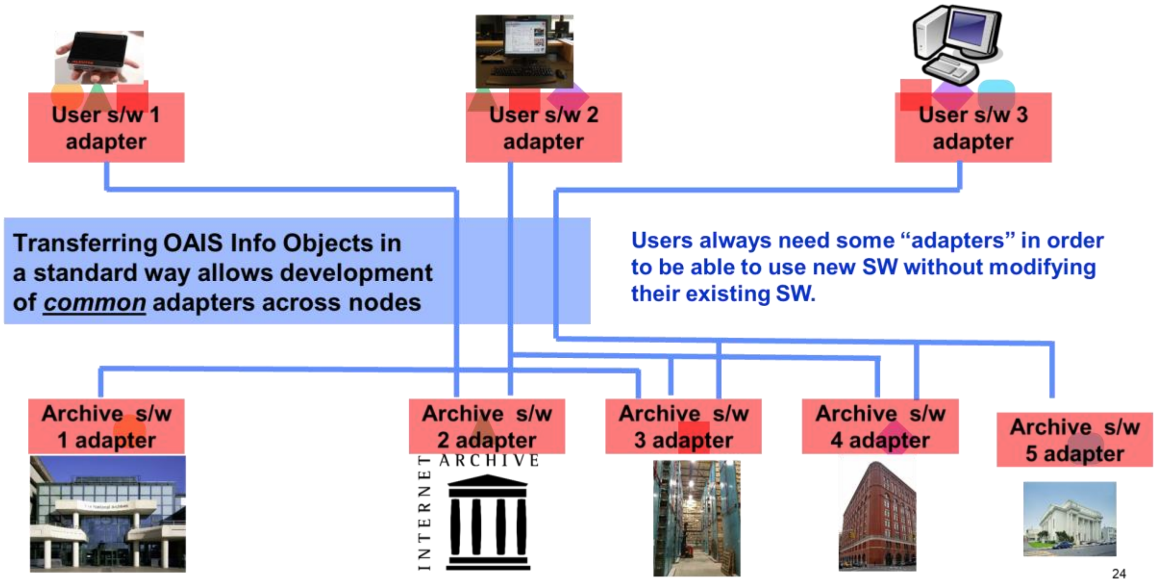
User s/w 1 adapter
User s/w 2 adapter
User s/w 3 adapter
Transferring OAIS Info Objects in a standard way allows development of common adapters across nodes
Users always need some "adapters" in order to be able to use new SW without modifying their existing SW.
Archive s/w 1 adapter
Archive s/w 2 adapter
INTERNET ARCHIVE
Archive s/w 3 adapter
Archive s/w 4 adapter
Archive s/w 5 adapter
24

Figure 2-3 Connections between clients and archives using OAIS-IF

NOTE: any of the objects defined in the OAIS Information Model may be requested. In general one would expect to receive either the object itself or an Identifier which allows the object to be obtained at a later time.

This has the advantage that each client only needs one adapter, specific to the client software being used, which allows that client to accept and use the OAIS-IF Information Objects, independent of the archive with which it is communicating, as long as each of those archives, with their adapters, use OAIS-IF Information Objects.

Of course, the various Information Objects will differ in detail, but they will all contain a Data Object plus the related Representation Information, or more specifically the first part of the Representation Information Network (RIN). Note that OAIS uses the term Representation Information both for a specific item such as a dictionary or piece of software, as well a RIN.

Therefore, the client with its single adapter can get, and understand as far as possible (given the required amount, quality or fidelity of the Representation Information), information from multiple archives simultaneously. As discussed in more detail below, use of OAIS-IF will make combining information from multiple sources easier, but cannot guarantee that it will be automatic or effortless.

An alternative way one might think to implement this would be to find a common format into which all other data could be transformed. However;

1. Format by itself is not adequate. OAIS uses the term Structure Representation Information as a general way of talking about "format", but it recognises that one also needs semantics (Semantic Representation Information") and "Other Representation Information" such as software.

2. Those formats which are extremely flexible such as HDF5[1] or HDS[2] require that the software which uses it is specifically tailored to the way in which the structure of these files is arranged.

[1] https://www.hdfgroup.org/solutions/hdf5/

[2] See http://starlink.eao.hawaii.edu/docs/sun92.htx/sun92.html

3. A common format has often been sought, and indeed many formats have the word "Common" or even "Universal" in their name. Moreover, experience shows that such transformations always lose information, and formats named "Common" or "Universal" fail to live up to their names.
4. The sheer size of the data may preclude such transformations in any practical way.

Therefore OAIS-IF is designed to follow the more general OAIS Information Model, so that OAIS Information Objects are transferred, hence explaining why this is called **OAIS**-IF. The advantage of using Information Objects is that they can contain any digitally encoded information but are required to have associated Representation Information, which allows the information encoded in the digital object to be understood – this is discussed in more detail below.

In order to provide a little more detail Figure 2-2 can be expanded as follows.

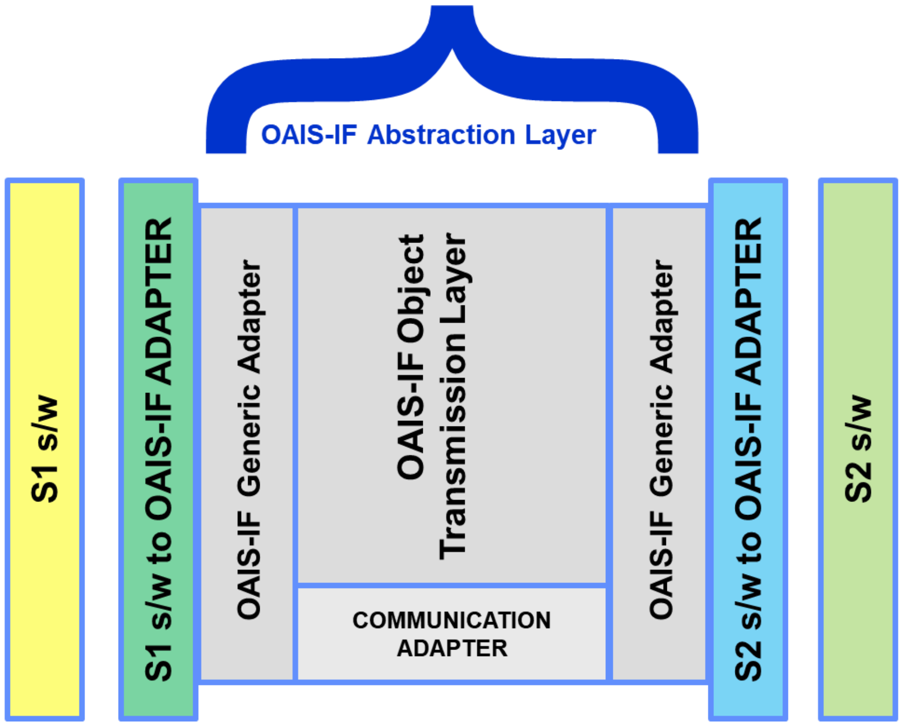
OAIS-IF Abstraction Layer
S1 s/w
S1 s/w to OAIS-IF ADAPTER
OAIS-IF Generic Adapter
OAIS-IF Object Transmission Layer
COMMUNICATION ADAPTER
OAIS-IF Generic Adapter
S2 s/w to OAIS-IF ADAPTER
S2 s/w

Figure 2-4 Expanded view of use of OAIS-IF

The figures shows that the OAIS-IF Abstraction Layer can be broken down into a "generic adapter" at each side of the OAIS-IF Object Transmission Layer. The latter, which uses a communication adapter to deal with network details, arranges largely for the transmission of OAIS Packaged Information Objects.

The S1 s/w is, for example, existing software which probably was created without knowledge of OAIS-IF.

In order to integrate with OAIS-IF one needs an adapter, called here the "S1 to OAIS-IF adapter" or "S1 specific adapter" i.e. the adapter is specific to the S1 software, for convenience.

The “S1 specific adapter” must be software tailored to the S1 software and must also be software which implements the OAIS-IF interfaces (shown here as the OAIS-IF Abstraction Layer) and required functionality.

The adapter by itself may not be able to use the data and Representation Information it initially receives, and therefore it requires additional Representation Information either from the same source or from different sources. It is also possible that there simply is not enough Representation Information available to make the information received usable by the S1 software. For example, the S1 software may only be able to display images and so will never be able to deal with scientific tables, unless the Representation Information essentially does something additional, such as displaying an additional window, independent of the S1 software’s image display, which can deal with such tables.

At the very least, the Representation Information may simply be pointers to documentation for the user to read, or pointers to software which is tailored to the data and which needs to be installed separately.

The adapter software includes parts which implement the communications between S1 and S2, either embedded within OAIS-IF or as a separate set of interfaces used by the S1 adapter (shown here as the Communications Adapter). Communications software already exists, and various frameworks provide convenient ways to support multiple communications stacks without changing the adapters.

In a symmetrical way the “S2 to OAIS-IF adapter” (or S2 specific adapter for short) must be software tailored to the S1 software and must also be software which implements the OAIS-IF interfaces and required functionality.

Of course, if S1 (or S2) already implements OAIS-IF then the associated adapter is not needed, but this special case does not cause any problems.

Note that arrangement of this diagram in layers is to show that each layer only communicates to its adjacent layers, through well-defined interfaces.

The layers do not specify **where** those layers are placed. Other diagrams will show potential placements of the software – see Figure 8-10.

2.3 EXAMPLE IMPLEMENTATION OF AUTOMATICALLY USABLE REPRESENTATION INFORMATION FOR MULTIDISCIPLINARY WORK

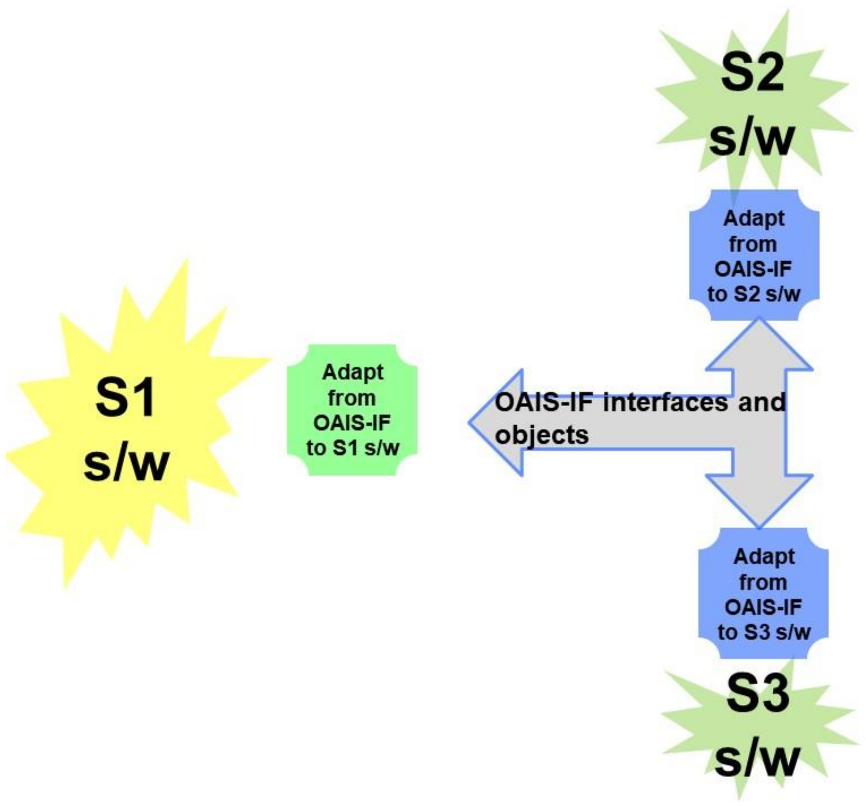
S2 s/w
Adapt from OAIS-IF to S2 s/w
S1 s/w
Adapt from OAIS-IF to S1 s/w
OAIS-IF interfaces and objects
Adapt from OAIS-IF to S3 s/w
S3 s/w

Figure 2-5 Multi-archive example

To provide a more concrete example which also show the use of the single adapters and the Representation Information, this illustrates a user's software S1 which gets information from two archives which use software S2 and S3.

One type of Representation Implementation which can be used quite easily is as serialized Java objects in a Java implementation. Such as object can be de-serialised to create a Java Object which can be examined through the Java Reflection API to discover which interfaces this object implements. This type of Representation Information is used in the example below.

For example, if one is using client software (S1) which can use JTable interfaces and which, via the adapter, receives information from an archive containing FITS files (S2). The information is in the form of an Information Object as defined by OAIS-IF.

Information may also be obtained from S3, with its own adapter. In this case the archive contains tables in the form of CSV files.

The adapter associated with S2 constructs and sends the Information Object. The adapter associated with S1 can separate the Data Object, which in this example is a FITS file which contains a FITS table, and Representation Information. The Representation Information in this example is a serialized Java object which, when de-serialised in the adapter, becomes a Java object. This Java object, as the adapter is able to discover, implements an interface which can read FITS tables and also implements the Java JTable interface. The adapter is tailored to the client software and therefore is built specifically to be able to provide the JTable objects to S1.

In a similar way the adapter for S3 constructs Information Objects consisting of a Data Object (the CSV file) and the associated Representation Information (the Structure RepInfo which described the UTF-8 and CSV encoding, and the Semantic RepInfo which describes the names, meanings and units of the columns plus Other RepInfo which in this case is again a serialised Java Object). As with the Information Objects from S2, the S1 adapter separates the Data Object from the Representation Information. De-serialising the Other RepInfo reveals that the Java Object also implements the Java JTable interface, using the Data Object, Structure RepInfo and Semantic RepInfo.

In exactly the same way as with the Information Objects from S2, the S1 adapter provides JTable Objects to S1.

Having JTables both from archives S2 and S3 the Information User of S1 can then combine, compare, etc., the information from these two sources.

In this way each of S1, S2 and S3 have their own unique adapters while the Representation Information allows the information to be combined for multidisciplinary work.

It can be seen that the same will apply whether S1 obtains Information Objects from any number of archives.

The separation of the adapters into generic and specific parts is shown in Figure 2-6, which includes an initial indication of functionality.

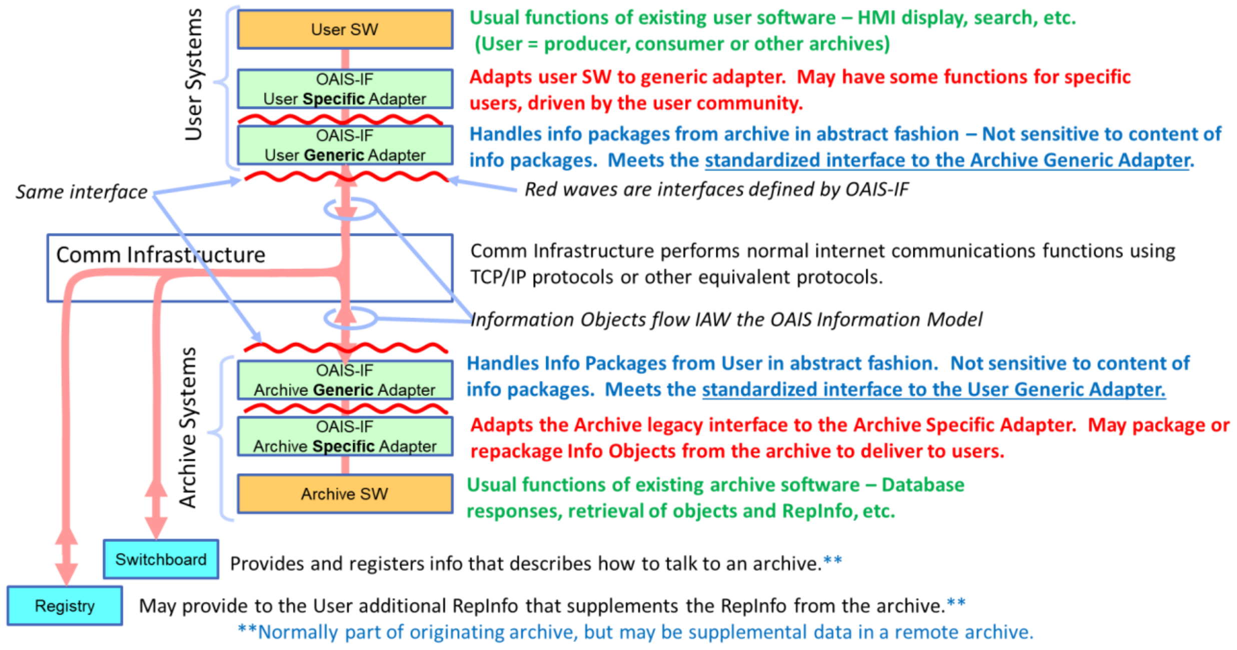
User Systems
User SW
OAIS-IF User Specific Adapter
OAIS-IF User Generic Adapter
Usual functions of existing user software – HMI display, search, etc. (User = producer, consumer or other archives)
Adapts user SW to generic adapter. May have some functions for specific users, driven by the user community.
Handles info packages from archive in abstract fashion – Not sensitive to content of info packages. Meets the standardized interface to the Archive Generic Adapter.
Same interface
Red waves are interfaces defined by OAIS-IF
Comm Infrastructure
Comm Infrastructure performs normal internet communications functions using TCP/IP protocols or other equivalent protocols.
Information Objects flow IAW the OAIS Information Model
Archive Systems
OAIS-IF Archive Generic Adapter
OAIS-IF Archive Specific Adapter
Archive SW
Handles Info Packages from User in abstract fashion. Not sensitive to content of info packages. Meets the standardized interface to the User Generic Adapter.
Adapts the Archive legacy interface to the Archive Specific Adapter. May package or repackage Info Objects from the archive to deliver to users.
Usual functions of existing archive software – Database responses, retrieval of objects and RepInfo, etc.
Switchboard
Provides and registers info that describes how to talk to an archive.**
Registry
May provide to the User additional RepInfo that supplements the RepInfo from the archive.**
**Normally part of originating archive, but may be supplemental data in a remote archive.

Figure 2-6 Indication of functionality in adapters

The red squiggly lines show where interfaces must be defined. It may be possible to define interfaces elsewhere.

The Switchboard has been introduced to provide information needed for communication between, in this case, users and archives. While the Registry provides additional Representation Information is required.

2.3.1 SIMPLE EXAMPLE OF A GENERAL ADAPTER FOR FILE-TYPE SERVERS

If one of the information sources is a file server, for example a web server, which provides the Data Object containing, in the particular case of a web server, HTML, HTTP and encoded images etc, then the adapter would have to construct an Information Object consisting of this Data Object with appropriate Representation Information. The Representation Information may be kept in a separate part of the web server and a simple look-up table could be used to identify which pieces of Representation Information to include.

This Information Object would then be sent in an Information Package to the receiver (S1 for example) using some agreed communications protocol, for example TCP-IP.

Other components of the Information Model may be handled in the same way. For example, the Data Object which encodes Provenance Information may be made up of a standard file, for example saying that all the web pages were constructed by a certain team of people, with additional text identified using a similar look-up table, including for example a script which extracts the version history of each page from the Content management System (CMS).

3 OAIS-IF SCENARIOS

The following provides a limited set of user scenarios which OAIS-IF will support.

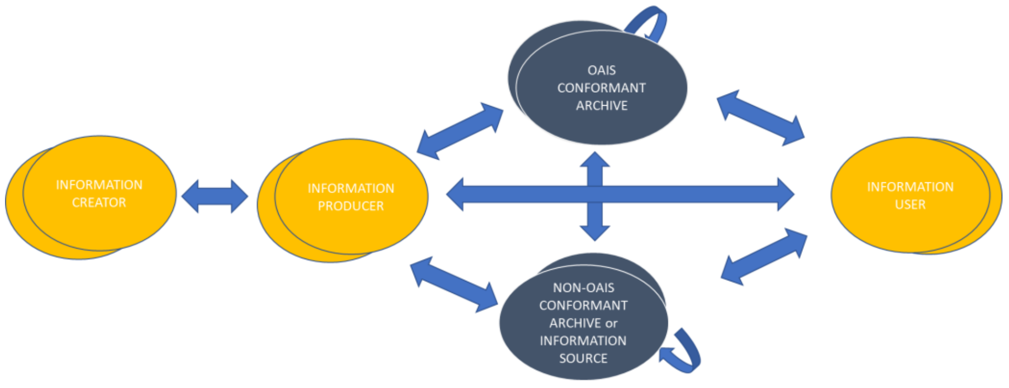

Figure 3-1 Roles of entities exchanging information

Information of various types is exchanged between individuals or organisations or systems which may be fulfil one or more of the following roles:

- Information Creators
- Information Producers
- Information Users, some of whom may be members of one or more Designated Communities of one or more OAIS conformant archives.

  NOTE: OAIS uses the term “Consumers” to refer to the role played by those persons, or client systems, who interact with OAIS services. Therefore, Consumers are a subset of Information Users because the latter may interact with services which are not connected with an OAIS.
- OAIS conformant Archives

  NOTE: Archive-to-archive interfaces (for both OAIS-conformant archives and non-OAIS-conformant archives) may be specified in this document (and accomplished by implementers) by using “Producer” and “Consumer” interfaces rather than by archive-specific interfaces for transactions between archives.
- Archives, or other Information Sources, which are not OAIS conformant.

In the OAIS-IF set of documents all these exchanges of information are assumed to be encoded in one or more Data Objects, with associated Representation Information. It is reasonable to propose that these are packaged together into OAIS Information Packages (Figure 3-2).

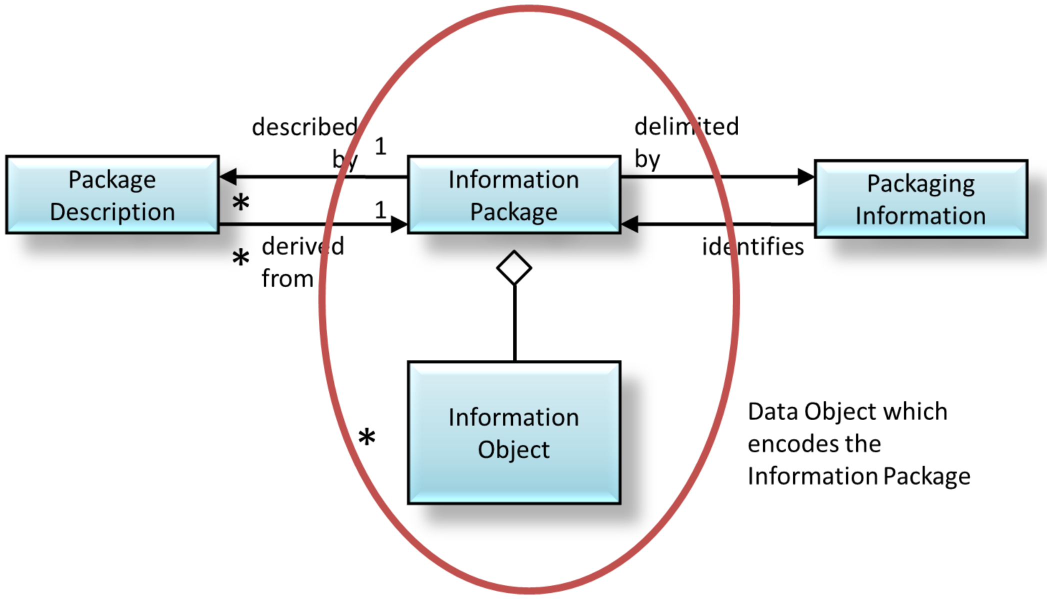

Figure 3-2 OAIS Information Package

The Information Packages are themselves made up of sets of bits, in other words Data Objects. These must be associated with their own Representation Information, of which, for the purposes of this document, Packaging Information is a part. The Information Packages will be transmitted between the entities by communication protocols such as TCP/IP, CCSDS DTN etc.

The full Archival Information Package (AIP) identifies all the pieces of information required to preserve the Content Information is shown in Figure 3-3. OAIS-IF should allow an implementation to deal with all these components of the AIP. However, this does not imply that all information sources, nor all transfers, use AIPs. The only requirement is that the design of OAIS-IF should support all the component objects required by an AIP.

For an OAIS Archive, the transferred Information Packages should have adequate content to allow an OAIS archive to build AIPs. For non-OAIS archives, other transactions (submission agreements, etc.) between the users and the archive will have to establish the needed content of the transferred Information Packages.

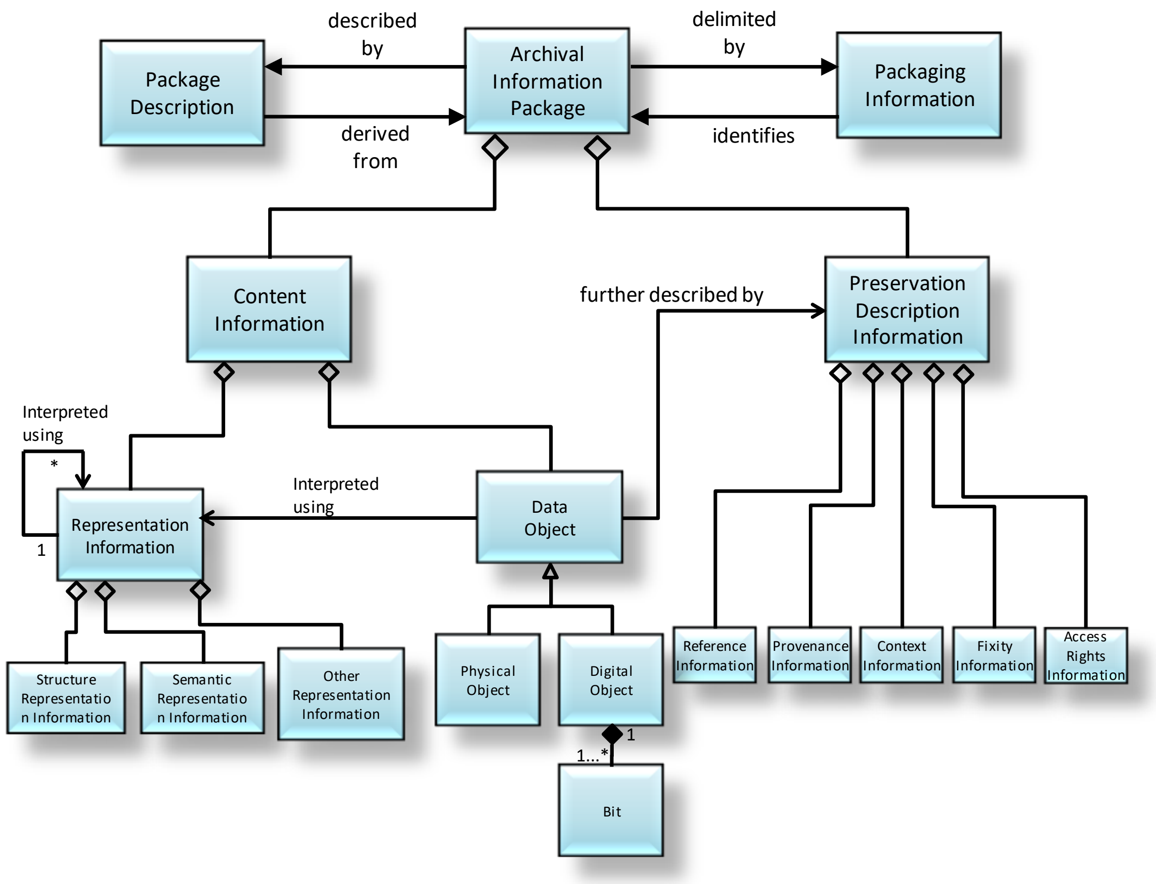

Figure 3-3 Detailed view of OAIS Archival Information Package (AIP)

Specific scenarios are listed next. In the following, for generality the term “User” can refer to a user, an archive (whether OAIS conformant or not) or an Information Creator or an Information Producer. The term “identifier” is used as a general term for something used to obtain the item required.

3.1 INFORMATION PRODUCER SENDS INFORMATION TO AN OAIS CONFORMANT ARCHIVE

Information is created by an Information Creator. It is sent to an OAIS Archive, via an Information Producer, which may be different from the Information Creator. OAIS defines, in general terms, a Submission Agreement that specifies the intended formats and content descriptions, and any other arrangements needed (such as delivery or transmission protocols), for the Data Submission Sessions. Such specifications establish the intended deliverables from the Producer and how they are represented on each delivery through physical media or in a telecommunication dialogue.

The Submission Agreement may be a document to be interpreted by a human. However, the extent to which these things can be computer interpretable is of interest for the OAIS-IF requirements.

The Information Producer may use the PAIS standard to encode the SIPs, in which case OAIS-IF may be used to define how the SIPs are transferred to the OAIS.

However, the Information Producer may use a different encoding for the SIPs, in which case OAIS-IF may be used to ensure that the SIPs may be understood by the OAIS.

Issues to be addressed include several which are mentioned as part of the Submission Agreement, but here the aim is to be more specific:

- How to arrange and carry out the transmission
- How to ensure that the Representation Information (Packaging Information) of the package is provided to the Archive.
    - If the Producer uses PAIS then the PAIS standard itself can be referred to, but if in the future there are multiple versions of PAIS then the correct version must be identified.
- How to ensure that the Archive can extract components from the Information Package Data Object.
- The OAIS Archive will expect to receive, perhaps over the course of several packages, the information required to create an AIP.

If a sequence of transmissions is to be arranged, with authentication mechanisms and fixity checks a context may need to be maintained for the Information Packages, for example to ensure packages are kept in sequence. Alternatively, the information may be provided in documentary form, rather than through an API.

3.2 INFORMATION PRODUCER SENDS INFORMATION TO A NON-OAIS CONFORMANT ARCHIVE

This scenario is similar to the one with the OAIS Archive. Significant differences are that there might not be a formal Submission Agreement and the archive might not require all the components of an AIP.

3.3 OAIS CONFORMANT ARCHIVE SENDS INFORMATION TO AN INFORMATION USER

In this scenario the user searches for the information he/she requires and the OAIS Archive sends back information as a response to the search and then the information requested in the form of one or more DIPs.

The following steps are involved:

1. The user inputs search criteria and identifies repositories which contain required information.
2. The user chooses one (or more repositories) and queries what is available.
    a. The user may need to log in to see what she/he is allowed to access.
3. The user then obtains the information from the repository.
    a. The user may simply get a copy of the whole AIP or
    b. The user may get a piece of information created by the repository suitably processing its holdings.

OAIS-IF treats queries and responses as messages in Information Packages. The DIPs are also Information Packages.

Since the Archive is OAIS conformant then according to one of the Mandatory Responsibilities a user who is a member of the Designated Community must be able to:

1) obtain enough Representation Information to understand/use the Data Object in the DIP and

2) obtain all the information related to Authenticity, and so the OAIS-IF must support this.

If the user is not a member of the Designated Community, then it does no harm to allow him/her to have the same functionality but in this case the Archive will not guarantee to be able to supply enough Representation Information to allow that user to understand/use the Data Object. This would be the case if the provided technical lexicon or jargon is not familiar to a consumer from outside the Designated Community.

3.3.1 AN INFORMATION USER WANTS TO OBTAIN INFORMATION FROM AN OAIS CONFORMANT ARCHIVE

1. User uses an Identifier for repository to search and select specific information to obtain an object identifier.
2. The user requests the object or part of an object.
3. The user receives the object requested.

3.3.2 AN INFORMATION USER WANTS TO OBTAIN INFORMATION FROM A NON-OAIS CONFORMANT ARCHIVE

1. User uses an "Identifier" for repository to search and select specific information to obtain an object identifier.
2. The user requests the object or part of an object.
3. The user receives the object requested.

3.3.3 AN INFORMATION USER WISHES TO GET AN AIP

1. The object identifier is confirmed as pointing to an AIP rather than any other object.
2. The Data Object of the AIP is retrieved.
3. The Representation Information, of which Packaging Information is a part, associated with the Data Object is retrieved.
4. The Package Description Information is retrieved if required.

3.3.4 AN INFORMATION USER WISHES TO GET INFORMATION DERIVED FROM AN AIP

1. The user uses the identifiers for one or more AIPs to obtain object identifiers for the components of the AIPs.
2. The user may request some operation to be performed on the components to create new pieces of Information.
3. The user uses that identifier to obtain identifiers for any components of that component for example if the original AIP is an AIC then identifiers for the component.
   AIPs can be obtained and then identifiers for the components of those AIPs can be obtained, and so on.

3.3.5 IF REQUIRED, INFORMATION USERS ARE AUTHENTICATED AND AUTHORIZED

1. User requests authentication.
2. The user provides the appropriate username/password or private key etc.

3.3.6 INFORMATION IS TRANSFERRED AS ONE OR MORE INFORMATION PACKAGES

1. The user requests some information using the object identifier that can be used to obtain local identifiers for the various components of the AIP, or other pieces of information.
2. The information source constructs the object to be transferred e.g.:

a. Extracting the components from internal storage such as a database or filestore.
b. Some components may have sub-components such as individual events relevant to Provenance.

3.4 NON-OAIS CONFORMANT ARCHIVE SENDS INFORMATION TO AN INFORMATION USER

This scenario is similar to the previous one except that the archive may not be able to guarantee to be able to supply Representation Information or evidence about Authenticity.

3.5 OAIS CONFORMANT ARCHIVE EXCHANGES INFORMATION TO ANOTHER OAIS CONFORMANT ARCHIVE

The previous scenarios involving an OAIS could apply here, with one of the OAISes acting as either an Information Producer or Information User. The special case is where the first OAIS should be able to send a copy of a complete AIP to the second OAIS, either as a single Information Package or as several related packages.

3.6 OAIS CONFORMANT ARCHIVE EXCHANGES INFORMATION WITH A NON-OAIS ARCHIVE

In this case either archive could play the role of Information Producer or Information User. The difference from the previous scenario is that complete copies of AIPs are unlikely to be sent from the non-OAIS conformant archive.

3.7 NON-OAIS CONFORMANT ARCHIVE EXCHANGES INFORMATION WITH ANOTHER NON-OAIS ARCHIVE

This scenario is similar to the previous one but, since non-OAIS archives may not actually have AIPs, copies of complete AIPs are not likely to be exchanged.

3.8 INFORMATION CREATOR SENDS INFORMATION TO INFORMATION PRODUCER

Information is created by an Information Creator. This may involve obtaining Information from other sources, where the Information Creator acts as an Information User.

It is sent to an Information Producer, which may be different from the Information Creator.

3.9 POSSIBLE REPINFO USE CASES

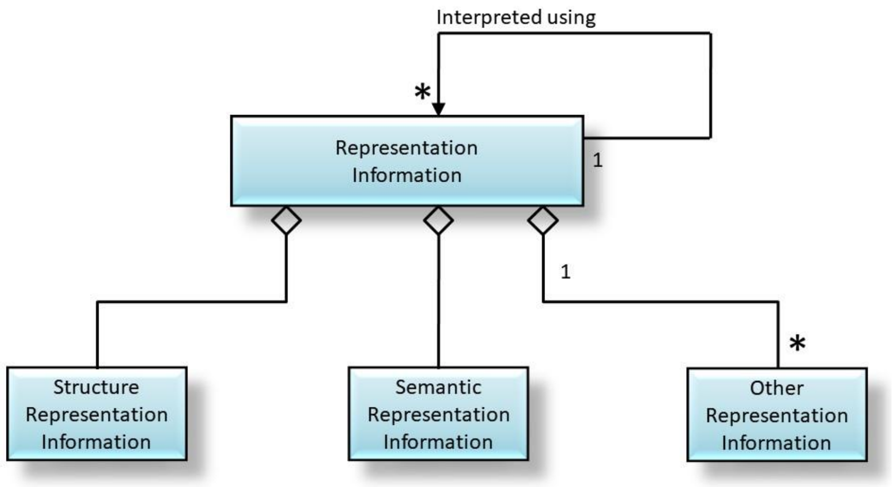
Interpreted using
*
1
Representation Information
1
*
Structure Representation Information
Semantic Representation Information
Other Representation Information

Figure 3-4 Fundamental types of RepInfo

1. Do something with a Data Object (DO) using RI e.g. as specified in the Preservation Objectives
    1. Identify the RI associated with the DO
    2. Use Semantic Information to identify an information element required
        1. Identify the Semantic Info components of the RI related directly to the DO (note that much of the RIN will relate to pieces of RI within the RIN)
        2. List the Semantic Info components of the RI related to the DO
        3. Are the Semantic Info components things which is understandable/usable by the user
    3. Use Structure Information to extract the specific information element
        1. Identify the Structure Info components of the RI related directly to the DO (note that much of the RIN will relate to pieces of RI within the RIN)
        2. List the Structure Info components of the RI related to the DO
        3. Are the Structure Info components things which is understandable/usable by the user
    4. Use Other RI
        1. Identify the Other RI components of the RI related directly to the DO (note that much of the RIN will relate to pieces of RI within the RIN)
        2. List the Other RI components of the RI related to the DO
        3. Are the Other RI components things which is understandable/usable by the user
    5. In addition one can see that there are other considerations:
        - There may be equivalent “versions” of an example of Representation Information e.g. software which may be written in C, C++ Java etc - any of

these may be used i.e. it may be (C-version OR Java version **OR** C++ version)

- Some types of Representation Information may be required e.g.
    - Structure Representation Information e.g a CSV text file **AND**
    - Semantic Representation Information e.g. a dictionary with the meaning, units etc of the table columns

3.10 POSSIBLE PROVENANCE USE CASES

1. Use Provenance Information to find out what has happened to a DO
    1. Identify the Provenance Information associated with a Data Object (DO)
    2. List provenance events in time order
    3. For each event, obtain (e.g. following PREMIS model): Object, Environment, Rights, Agent, Event - more specifically
        1. Data Object Identification Information (Reference Information??)
        2. Information about systems used
        3. Access Rights Information
        4. Agent Identification Information
        5. Event Description Information

3.11 SEARCH

Very little has been written here about SEARCH capabilities, although it is clear that there MUST be such capabilities to all the user to identify the information to request. OAIS has very little guidance on search capabilities. All data/information providers will have their own search capabilities, but each one will have its own search parameters language and search capabilities.

One can identify a number of general search mechanisms which could work for any information source and could also be used to identify data sets which could be combined.

1. Get a simple list with a description of all information available, which can then be used in a simple text search.
2. Full text search of the contents of all the data objects
3. Treat the search capability as encoded in a Data Object which has Representation Information to enable the user to user that Data Object in order to query the source e.g. an archive.

3.12 EARLIER USE CASE WORK

See https://www.dropbox.com/s/zx3gl35dq1vqgf6/UseCases_jsh_190110.xlsx?dl=0 , https://www.dropbox.com/s/r3jf3ticr86vgx3/UseCase_Service_Mapping_191014.xlsx?dl=0 and https://www.dropbox.com/s/59787kz3ttcwrl8/OAIS-IF%20Use%20Cases%2020181105.docx?dl=0 for earlier work on use cases.

4 OAIS-IF REQUIREMENTS

The term “archive” is used for a general information repository, whether conformant to OAIS or not; the term “OAIS” is used where the archive is OAIS conformant.

The following (incomplete) list of requirements for APIs have been identified so far. Requirements are not placed on the internal workings of the various entities, however some requirements may imply functionality from the archive, but if this functionality is not available then a NULL response may be made, and will be acceptable.

Table 1 Requirements for OAIS-IF - TO BE COMPLETED

REQUIREMENTS	SCENARIO
Req 1) APIs, based on, but not restricted to, the OAIS Information Model will be available.	3.1, 3.3, 3.5, 3.3.1, 3.3.6
Req 2) APIs for negotiations to allow additional Representation Information to be provided, if available, will be available.	3.3
Req 2-1) API to identify the category of a piece of RI e.g. Structure, Semantic, Other, and ideally more detailed categories.	
Req 2-2) API to obtain the full RIN or the immediately related pieces of RI	
Req 3) An API which allows a negotiation on transformations of the Data Object before transmission, will be available.	
Req 4) An API which allows the parties to agree on or discover a communications protocol to use will be available.	3.1
Req 5) An API to allow the archive to be able to verify that a user requesting access to the archive is authorized will be available.	3.3.5
Req 6) An API to allow the archive to be able to accept SIPs will be available, in particular	3.1
Req 6-1) Allow the archive to verify that Producer requesting access to the archive is authorized.	3.1
Req 6-2) Make available the definitions of the types of submission packages that the archive will accept.	3.1
Req 6-3) one or more interfaces which can be used to submit an SIP and reports back on status of the ingest	3.1
Req 7) APIs to support search of the archive to provide an identifier for required information will be available.	3.3
Req 8) APIs to retrieve information, as Packaged Information (i.e. Information Packages with Representation Information to allow the package to be unpacked), given one or more identifiers will be available.	3.3.1
Req 9) APIs to allow the discovery of communication options, e.g., how a client can communicate with an archive, should be available	3.1

5 MORE DETAILED CONSIDERATIONS

The approach may be carried out through the following stages.

5.1 BASIC OAIS INTERFACES

Construct Interfaces for each OAIS Information Model (IM) object, with minimal details. For example

1) an Information Object must have a Data Object and Representation Information, therefore there should be get/put methods for each of these.
2) Representation Information often comes in groups which include “Structure”, “Semantic” and “Other” categories of RepInfo. Therefore it would make sense to include the following:
    a. Any piece of RepInfo may have a category which at the very least could be a String so that an application could make an informed choice as to which piece of Representation Information to proceed with.
    b. The group of RepInfo could be one in which ALL the individual pieces of RepInfo must be used, alternatively the group may consist of alternatives so that a choice may be made to select the SINGLE piece of use e.g. software may be implemented as JAVA or C; these alternatives would be in a group from which one may be selected.
3) An Information Package will have get/put methods for the following objects, all of which are Information Objects
    a. Information Object
        i. If this includes PDI then this could be treated as a AndGroup, and in particular if the package is an AIP then all the components must be there.
    b. Package Description
    c. Packaging Information
4) An Archival Information Package will extend the Information Package by adding that the Information Object MUST include
    a. Data Object and its
        i. Representation Information
        ii. Preservation Description Information, which itself contains the following Information Objects:
            1. Provenance Information
            2. Context Information
            3. Reference Information
            4. Fixity Information
            5. Access Rights Information

    Note that the any Information Package may contain these objects, but there is no guarantee that they are complete from the point of view of the repository. Therefore, it would be sensible to add a method e.g.

Boolean iscomplete() which is TRUE when the AIP is believed to be complete and false otherwise.

5) Identifiers for objects must be available. These would allow more generality that simply a URI etc. The Identifier interface might have methods such as:
   - String toString() - returns a string

5.2 SUPPLEMENTS NEEDED TO OAIS IM INTERFACES

1) The OAIS IM does not provide any additional detail about object interfaces. All one can say is that any of the Information Objects implement these get/put methods. The implication is that one relies entirely on the Representation Information to be able to use/understand the Data Object in some way TBD, for example if the Identifier for the Representation Information is recognised and some existing software may be used to deal with the Data Object for that object. For example a Web browser can use the Media Type (aka MIME Type), as Structure RepInfo, to determine which pre-configured application to use.
2) Identifiers for objects must be available. These would allow more generality that simply a URI etc. The Identifier interface might have methods such as:

   toString() - returns a string

   toURI() – return URI if possible
3) The set of Interfaces would not provide constructors for any object – they must be provided by the implementing classes. Constructors could include:
   a. InformationObjectImpl() which creates a trivial class which implements the Information Object interface into which the put methods can insert the Data Object and RepInfo Object.
   b. InformationObjectImpl(DataObject identifier1, RepresentationInformation idendifier2), which essentially performs the "put"s in one step.
4) Authentication and Authorisation must be provided by the implementing classes.
5) The OAIS IM does not provide any guidance on search interfaces. A number of possibilities could be considered.
   a. We may define a simple interface such as
      i. search(String query) which return a list of identifiers. However, the structure of the string must be defined somehow.
   b. The search may be performed using a repository specific method – perhaps using the "Hollywood Principle"

5.3 FUNCTIONALITY FOR INTEROPERABILITY

Some specific functionality is required for interoperability including:

- Negotiate – to agree how to agree a strategy for communication and usability
- Get more RepInfo – request and receive RepInfo if required for usability

This functionality involves behaviour which cannot be described only by interfaces, instead other UML diagrams are needed including Sequence diagrams, Component diagrams and Deployment diagrams (section 7).

5.4 COMMUNICATION PROTOCOLS

In order to support interoperability, the communication protocols must be defined.

5.4.1 REST COMMUNICATIONS

REST (https://restfulapi.net/) is well defined as a way to communicate.

JSON (https://www.json.org/json-en.html) is widely used.

5.4.2 POSSIBLE JSON EXAMPLES

This is a simple example for a InfoPackage and an InfoObject, which should be understandable. The only non-OAIS native part if what is called the “AndGroup” and “OrGroup” with the example of the OtherRepInfo where there are 2 separate implementations – C# and Java – and one can choose which to use depending on which programming language is preferred.

In this example all the separate parts are referred to by URI, but one could instead put in simple text or an encoded binary object with something like binhex.

```
{
        "InformationPackage":{
                "PackageType":"AIP",
                "PackageDescription": "This  is an example AIP",
                "Provenance":{"IdentifierType":"URI", "Identifier":http://myprov.example.com/prov},
                "Reference":{"IdentifierType":"URI", "Identifier":http://myprov.example.com/ref},
                "AccessRights":{"IdentifierType":"URI", "Identifier":http://myprov.example.com/ar},
                "Context":{"IdentifierType":"URI", "Identifier":http://myprov.example.com/context},
                "Fixity":{"IdentifierType":"URI", "Identifier":http://myprov.example.com/fix},
                "RepInfo":{"AndGroup":[
                                        {"RICategory":"SemanticsRI", "IdentifierType":"URI", "Identifier":http://myprov.example.com/risem},
                                        {"RICategory":"StructureRI", "IdentifierType":"URI", "Identifier":http://myprov.example.com/ristr},
                                        {"RICategory":"OtherRI", "OrGroup": [
                                                    {"RICategory":"OtherRI", "IdentifierType":"URI", "Identifier":http://myprov.example.com/rioth-java-sw},
                                                    {"RICategory":"OtherRI", "IdentifierType":"URI", "Identifier":http://myprov.example.com/rioth-csharp-sw}
                                                            ]
                                          }
                                      ]
                            }
}
{
        "InformationObject":{
                "DataObject":{"IdentifierType":"URI", "Identifier":http://myprov.example.com/do},
                "RepInfo":    {"IdentifierType":"URI", "Identifier":http://myprov.example.com/ro,}    //This points to an RI Group */
        }
}
```

6 OAIS-IF DOCUMENT STRUCTURE

The following subsections outline documents in the OAIS-IF set, with notes on their contents.

Note: The following diagrams are drafts and will be updated in due course – see https://www.dropbox.com/s/c3ribtzpyz6ynh9/OAIS-IF-diagrams.pptx?dl=0 (Shared Dropbox: \CCSDS-DAI-Shared\Draft Documents\OAIS-IF Architecture Description\ OAIS-IF-diagrams.pptx)

Note that there may be name changes to the documents, and there may (temporarily) be inconsistencies in the set of documents defined.

6.1 OAIS DOCUMENT TREE

The following figure describes the OAIS Document Tree. The box colors follow the book color conventions of CCSDS: Magenta, Green and Blue.

See Figure 6-1. Particularly note the division between documents in the OAIS Process Framework (OAIS-PF)_and the OAIS Interoperability Framework (OAIS-IF). This document provides rationale, scenarios and requirements for the OAIS-IF. The OAIS-PF is shown as reference only. Acronyms are listed in section 1.6.1.

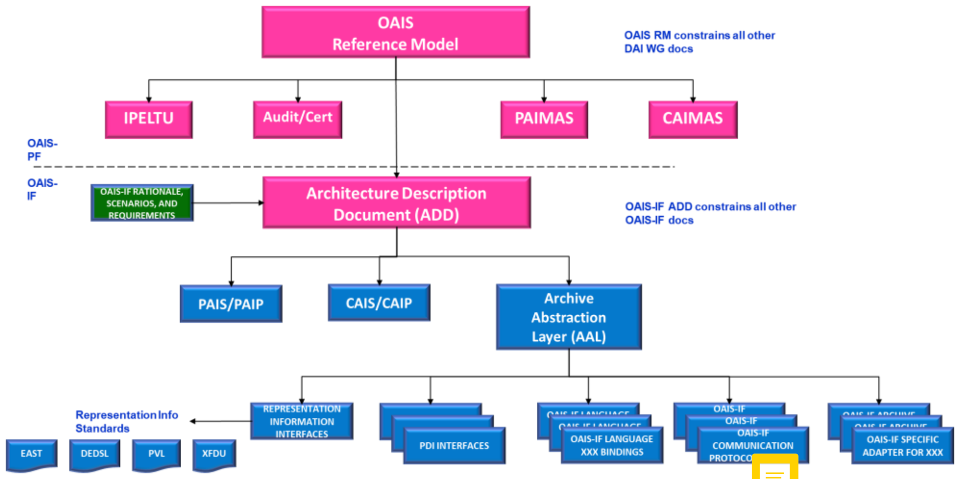
OAIS Reference Model
OAIS RM constrains all other DAI WG docs
IPELTU
Audit/Cert
PAIMAS
CAIMAS
OAIS-PF
OAIS-IF
OAIS-IF RATIONALE, SCENARIOS, AND REQUIREMENTS
Architecture Description Document (ADD)
OAIS-IF ADD constrains all other OAIS-IF docs
PAIS/PAIP
CAIS/CAIP
Archive Abstraction Layer (AAL)
Representation Info Standards
EAST
DEDSL
PVL
XFDU
REPRESENTATION INFORMATION INTERFACES
PDI INTERFACES
OAIS-IF LANGUAGE XXX BINDINGS
OAIS-IF COMMUNICATION PROTOCOL
OAIS-IF SPECIFIC ADAPTER FOR XXX

Figure 6-1: NOTIONAL OAIS/OAIS-IF Document Tree

6.2 OAIS-IF RATIONALE, SCENARIOS, AND REQUIREMENTS GREEN (THIS DOCUMENT)

- An overview of the architecture, what problems we believe the set of standards is to address, what it will cover and what it will not cover (we had lots of discussion of that) and how it relates to OAIS and the other DAI (and other) standards.
- What people/systems will be able to do which they cannot do now
- Some simple use cases to illustrate what we are addressing.
- Make it clear that we solve the central problem of how to deal with "bags of bits" in a general way applicable throughout the architecture.
- An indication of what Information Producers, Information Users and Archives will need to change to benefit from these standards.

6.3 ARCHITECTURE DESCRIPTION DOCUMENT (ADD) MAGENTA

- An overview of how we think things will actually fit together e.g., Steve's overall diagram plus some of the other UML diagrams (Class, Sequence, Component etc) which show different parts of the system in more detail.

See https://www.dropbox.com/s/my91nnl1o8qplfo/02_White_Book_Recommended_Standard_OAIS-IF_Draft_191216%20v5.3%2006_ComparedCombined_04_05_201222b_Cleaned.docx?dl=0 and https://docs.google.com/document/d/14V0wN6nEnG2MaSMmNClRzuDrTxcB0zjv0XNw2pdYDI8/edit#

6.4 OAIS-IF ABSTRACTION LAYER (OAL) BLUE

This is a Blue Book because it is implementable and there should be 2 interoperable implementations from NASA and ESA.

To publish a protocol interface spec, it should define
- actual PDUs,
- info models,
- behavior,
- peer to peer interactions in some unambiguous form

- The directly OAIS related Information Model interfaces, adding in methods and additional interfaces such as Identifiers etc.
- The OAIS Unique Selling Points of understandability
    - Negotiation interface and
    - a minimal interface for Registry and Switchboard
- How we pass information between components e.g., Packaged Information

    - Including PDU e.g.
        - REST: HPPT(S), JSON serialisation and HTTP responses e.g. http://swagger.io
        - XML schema for SOAP
- Some specific interfaces for Representation Information, Provenance, Fixity, Access Rights, Reference (which may link to Identifiers) if we actually want interoperability. With modern programming languages we do not need to say that these are the only possible interfaces for these specific types of Information, but we recommend these.
- The Generic Adapter Interfaces and functionality will be defined, including the Generic Adapter to Generic Adapter and the Generic Adapter to Specific Adapter interfaces.

See http://int-platform.digitalpreserve.info/wp-content/uploads/sites/5/2014/12/javadoc/model/ and http://int-platform.digitalpreserve.info/wp-content/uploads/sites/5/2014/12/javadoc/framework/ for an example of interfaces from an old project.

6.5 OAIS-IF ARCHIVE ADAPTERS MAGENTA

- What Archives need to have in order to interact – how to get from what the archives can provide to what the OAL demands?
- Example Specific Adapters e.g. for systems which normally simply provide a Data Object such as a science data file or a web page.
- What users need to have in order to be able to interact. This could be how to deal with encoded information in general terms plus some specific examples such as images, tables and trees.

6.6 OAIS-IF JAVA BINDING (BLUE BOOK)

- How to implement the OAL in Java (should be fairly straightforward)

6.7 OAIS-IF XXX BINDING (BLUE BOOK)

- How to implement the OAL in language XXX (difficulty depends on the language)

6.8 OAIS-IF COMMUNICATION PROTOCOL BINDING (BLUE BOOK)

- How to use existing communication protocols, including CCSDS, TCP/IP, SPRING, etc

6.9 OAIS-IF SPECIFIC ADAPTER FOR XXX (BLUE BOOK)

- Definition of specific adapter interfaces to interact with software XXX.

6.10 PRODUCER-ARCHIVE INTEROPERABILITY PROTOCOL (PAIP) (BLUE BOOK)

- SIPs and how to transfer them using:
    - PAIS
    - OAIS-IF documents above

- Notes on “out of band” communications to make arrangements.

6.11 CAIS/ PRODUCER-ARCHIVE INTEROPERABILITY PROTOCOL (CAIP) (BLUE BOOK)

- Queries – Use variety of existing standards
- DIPs

Figure 6-2 illustrates the use of the various components described by the documents described in this section, where the client software (e.g. PDS, Astronomy or EO software tools) uses an Adapter with a specific programming language binding to create objects specified by the OAL. The Archive adapter, with a specific programming language binding converts the OAL objects into ones which the Archive Native Interface can use to obtain the required information from the Archive Internals.

These objects are communicated through the Communications Adapter (shown here as being based on SPRING, REST or TCP, but others are possible). It is shown as a vertical bar to indicate that there are alternative ways of distributing the various pieces of software..

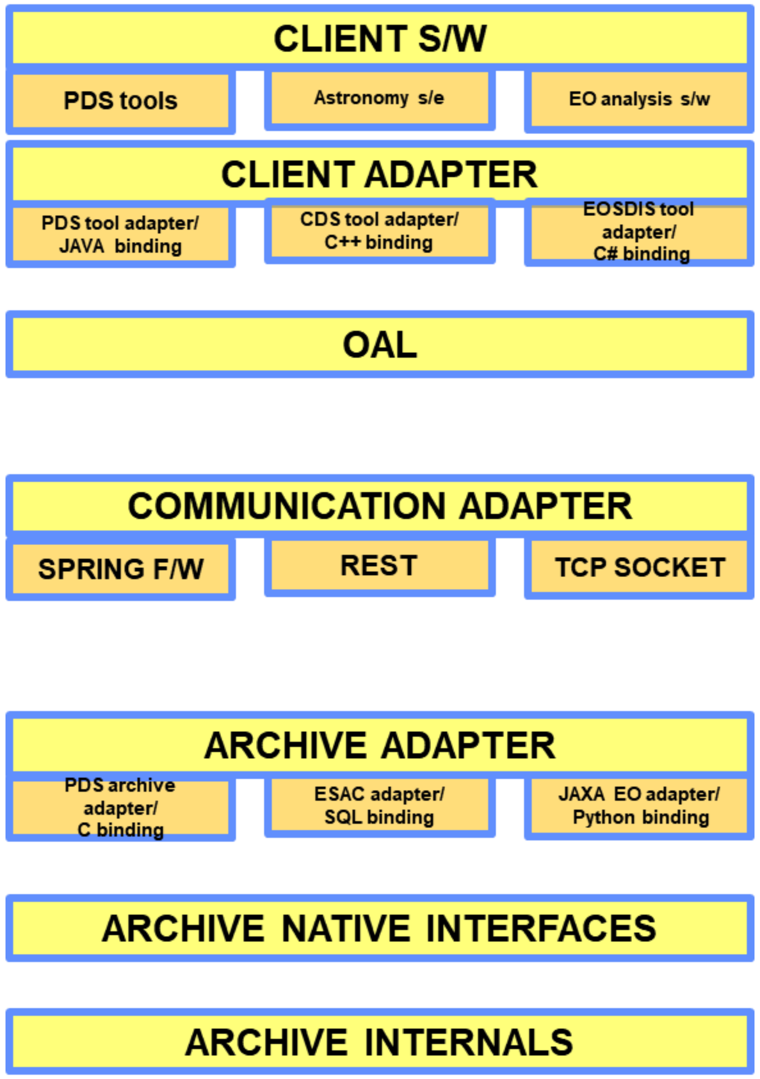
CLIENT S/W
PDS tools
Astronomy s/e
EO analysis s/w
CLIENT ADAPTER
PDS tool adapter/ JAVA binding
CDS tool adapter/ C++ binding
EOSDIS tool adapter/ C# binding
OAL
COMMUNICATION ADAPTER
SPRING F/W
REST
TCP SOCKET
ARCHIVE ADAPTER
PDS archive adapter/ C binding
ESAC adapter/ SQL binding
JAXA EO adapter/ Python binding
ARCHIVE NATIVE INTERFACES
ARCHIVE INTERNALS

Figure 6-2 Illustrative software stack

7 OAIS-IF MORE DETAILED UML ANALYSIS

7.1 ASSUMPTIONS

1) We are focussed on information that comes out of and goes into things like OAISes and Applications (Producers/Consumers/Management).
2) We are concerned with OAIS functions within the archive **only** when they have characteristics that surface at user interfaces (users including other archives).
3) The information we are specially qualified to facilitate the exchange of is that which is related to OAIS Information Model.
4) The OAIS Information Model should form the basis of our API/protocol/binding/plugin, but things need to be added e.g. security.
5) We should not be concerned about optimisation right now.
6) We want to facilitate the exchange of information between disciplines and OAISes so it can be used and/or preserved.
7) An Information Package is a Data Object; its Packaging Information is its RepInfo which allows one to unpack the package and identify its components.. Together they make an Information Object.

The way to deal with an Information Object is the following

How to deal with an IO – fundamental ideas

a) Each Information Object (IO) must have an ID which identifies its RepInfo, an Object RepInfo ID (ORIID)

b) Given the ORIID the receiver of the IO can do one of the following:
   a. Use the IO because the receiver knows how to deal with things that have that ORIID
      OR
   b. Negotiate with the sender to see if it can send an alternative “form” which it does know how to use e.g. send a list of ORIIDs which it knows how to deal with and see if the sender can transform the Data Object to a preferred ORIID.
      1. If the negotiation is successful then (a) applies **OR**
      2. Use the ORIID to get as much of the RIN (RepInfo Network) from a registry as it needs to use the DO.
         a. The receiver may negotiate with the registry to get RepInfo it can use e.g. is there some Java software that can use the Data Object in the IO?
            i. To help decide which piece of the RIN to use, the RIN must have some mechanism for indicating which role the next pieces of RepInfo play e.g. Semantic, Structure or Other.

This is used throughout the application of the OAIS-IF.

7.2 DESIGN IDEAS

We want to design the system in a way that makes implementations easier and more widely usable.

1. For example if we can design software in a way that we can re-use the same piece of software in lots of different places in the system then that is a simplification because we don’t need to write so much software.
2. On the other hand we may realise that we need to change the software to deal with different circumstances e.g. we might want to use the system with different types of networks, or with different sources of information or different ways of putting information together, or using different programming languages. If we break the design into layers we stand a better chance of isolating these various dependencies so that, for example, we can replace the software that deals with one network (say

TCP/IP) with that which deals with another network (such as some CCSDS protocols), without changing anything else in the system.

3. Another simplification is whether we can re-use some existing design/software which itself helps with (1) and/or (2)
4. Existing software, which is specific to an Archive, may be used to access information.

The ideas here are aimed at designing the system to be able to re-use (see (1)) some pieces of software to deal with understandability – which pops up everywhere unless what we send is very rigidly specified.

The CCSDS MAL speaks to (3) above, but there are lots of other possibilities. The use of the JAVA repository and registry API would also be in line with (3).

For point (2) there are lots of possibilities and design choices. The discussion about, say, an API for Provenance was to provide a proposal to address part of this. Different Producers, Consumers and Archives also strongly suggest that we try to define layers where they are obvious now which isolate the specifics of each while keeping the rest of the system unchanged. Other layers may be identified later, perhaps breaking an initial layer into sub-layers.

UML diagrams support the development of designs to support these ideas.

7.3 (ARBITRARY) DECISIONS

Send information in OAIS Information Packages

- This allows us to send any information, so we can re-use software in many places.
- If a package goes into an OAIS we can call it an SIP
- If a package comes out of an OAIS we can call it a DIP
- If the package goes to or comes from a place other than an OAIS, then we just call it a package.
- If a package has everything needed for preservation of the (content) Information then it can be called an AIP.
  - Whether or not it actually has everything needed for preservation depends upon the Designated Community of the AIP, as defined by the OAIS, hence we have to decide:
    1. The Package needs enough information about the Designated Community in order to decide whether there is enough RepInfo etc

    OR

2. The specific OAIS can declare that this Package is an AIP e.g. by setting a flag
   - This is probably the easiest way to go

One very convenient type of RepInfo is software that can be used in an application. Here are some proposed standard APIs

For an Information Object, software should support an interface:

- getDataObject (returns a DataObject – perhaps as ID (DOID))
- getRepInfo (returns RepInfo – perhaps as ORIID)

For an Information Package, software should support an interface:

- - getPackagingInfo (returns a ORIID)

The PackagingInfo has the following interface:

- getDataObject (returns a DataObject – perhaps as ID (DOID))
- getRepInfo (returns RepInfo – perhaps as ORIID)
- getProvenanceInfo (return ProvenanceInfo – perhaps as DOID - or NULL if not there)
- getReferenceInfo (return ReferenceInfo – perhaps as DOID - or NULL if not there)
- getAccessRightsInfo (return AccessRightsInfo – perhaps as DOID - or NULL if not there)
- getContextInfo (return ContextInfo – perhaps as DOID - or NULL if not there)
- getFixityInfo (return FixityInfo – perhaps as DOID - or NULL if not there)
- isAIP(return true or false)

For each of these Info Objects we can provide a default API e.g.

ProvenanceInfo (see https://docs.google.com/document/d/1YCyhBZKRP7IWhArdI3MnwahjZY5K93UsDZq-cWApYeI/edit?usp=sharing)

Contex

- getContext (returns an Information Object which contains text which provides the context)

ReferenceInfo

- getURI()

AccessInfo

Get????

FixityInfo

- getFixity(????)

RepInfo

8 UML DIAGRAMS

UML defines a number of different diagram types, each designed to show particular aspects of the system as any one particular diagram cannot show everything.

The following diagrams are similar, but not identical, to the text above.

8.1 USE CASES

The Use Cases shown in the diagram focus on the interactions between Information Users and Archives. The Information Producers are largely ignored here because

1. The Producer-Archive interactions are more complex
2. There are typically many fewer Producers than Consumers for any specific Archive,
3. Typically Producers would interact with a small number of Archives whereas Consumers would tend to interact with a large number of Archives.

Consumers will interact with familiar Archives, getting Information with which they are familiar (although this may not apply to Provenance Info etc). We show here that such Consumers use the Native Access Methods provided by the Archive. These Consumers will already know how to use the Information held by the Archive, and indeed they may be members of the Designated Community.

What we wish to focus on here are the interactions with other Consumers, those for which the Archive is unfamiliar. In this case we will need to provide extra Representation Information.

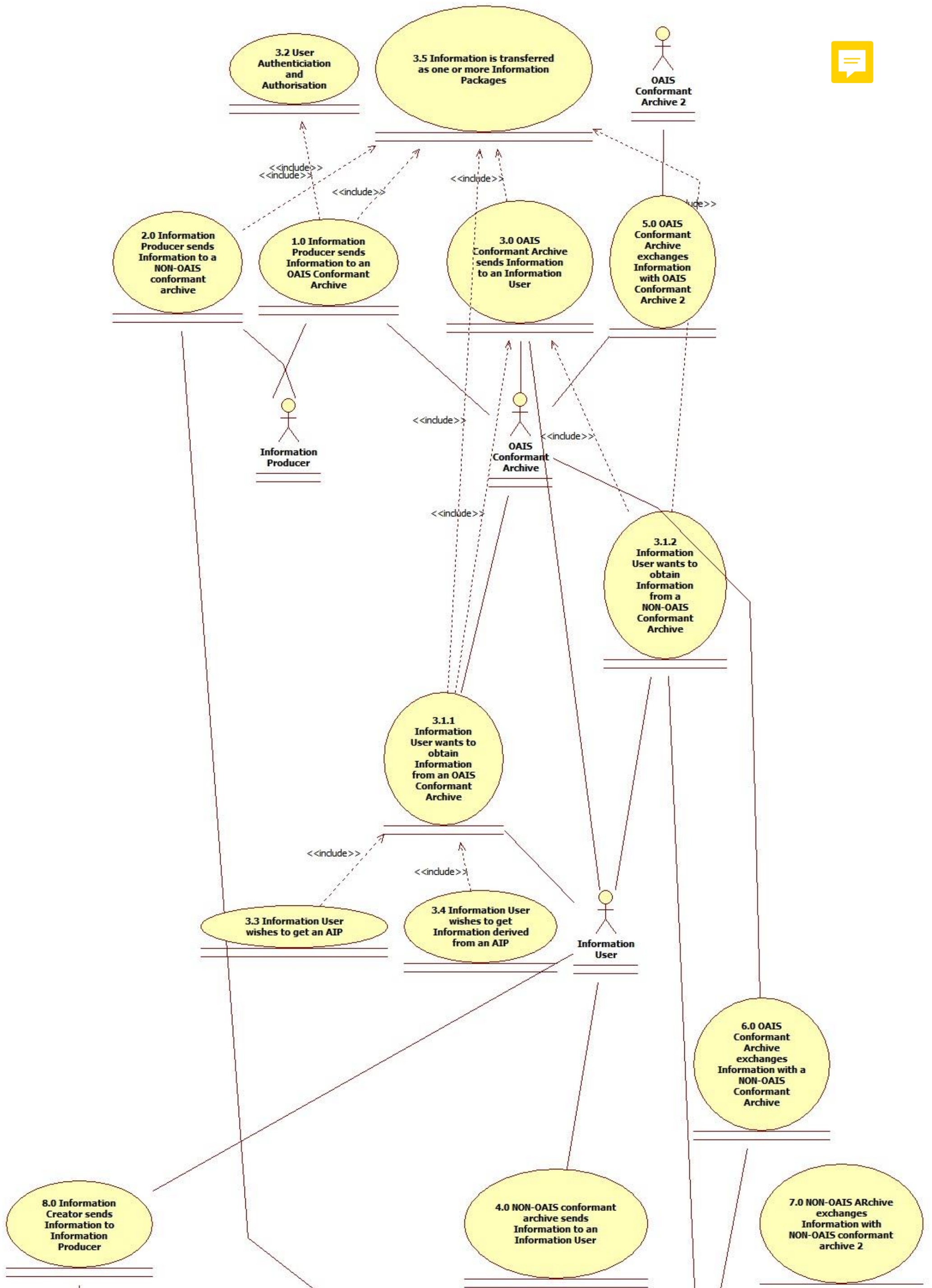
3.2 User Authenticiation and Authorisation
3.5 Information is transferred as one or more Information Packages
OAIS Conformant Archive 2
<<include>>
2.0 Information Producer sends Information to a NON-OAIS conformant archive
1.0 Information Producer sends Information to an OAIS Conformant Archive
3.0 OAIS Conformant Archive sends Information to an Information User
5.0 OAIS Conformant Archive exchanges Information with OAIS Conformant Archive 2
Information Producer
OAIS Conformant Archive
3.1.2 Information User wants to obtain Information from a NON-OAIS Conformant Archive
3.1.1 Information User wants to obtain Information from an OAIS Conformant Archive
3.3 Information User wishes to get an AIP
3.4 Information User wishes to get Information derived from an AIP
Information User
6.0 OAIS Conformant Archive exchanges Information with a NON-OAIS Conformant Archive
8.0 Information Creator sends Information to Information Producer
4.0 NON-OAIS conformant archive sends Information to an Information User
7.0 NON-OAIS ARchive exchanges Information with NON-OAIS conformant archive 2

Figure 8-1 Use Case diagram

8.2 SEQUENCE DIAGRAM

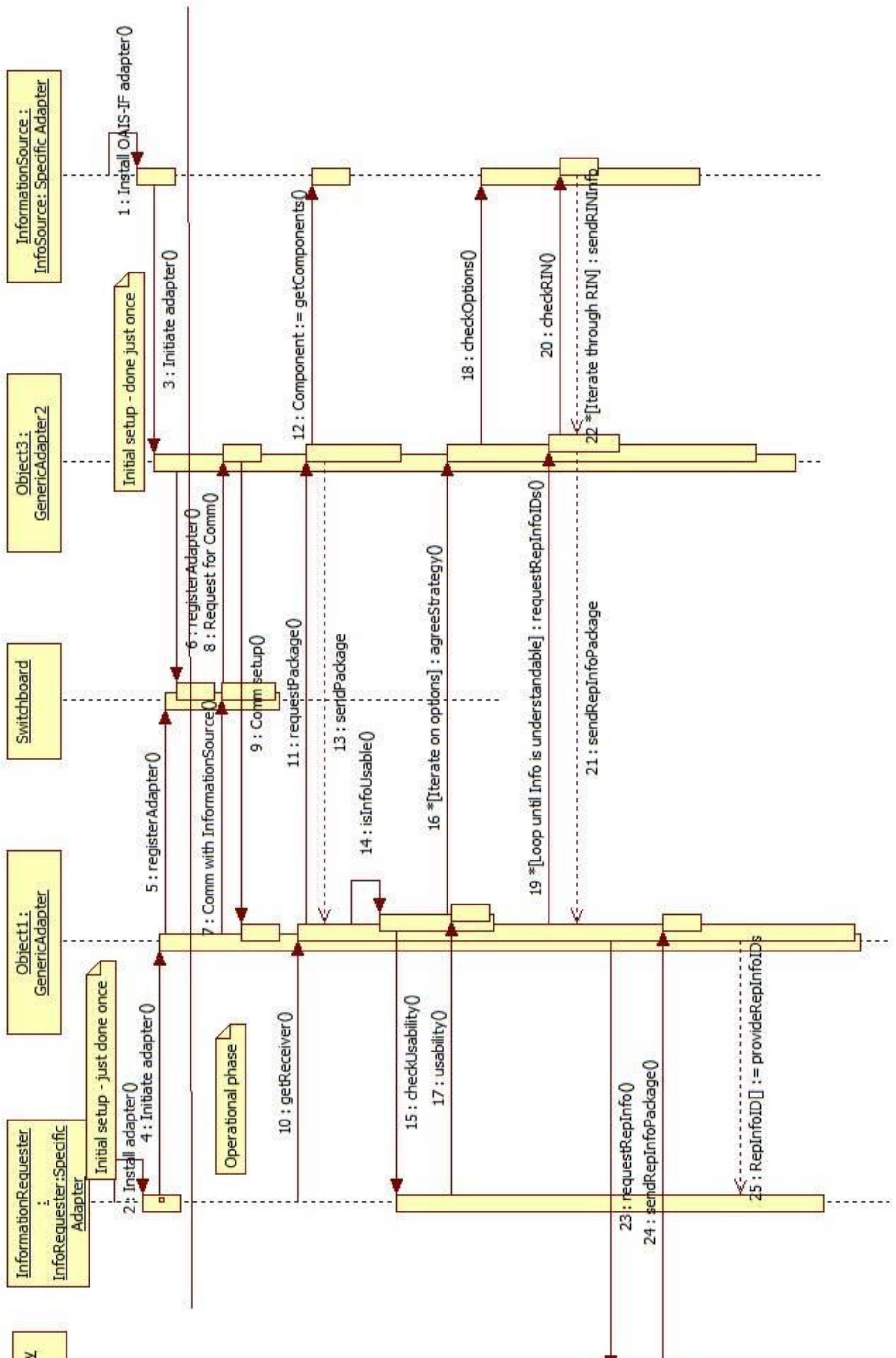
InformationSource : InfoSource: Specific Adapter
Object3 : GenericAdapter2
Switchboard
Object1 : GenericAdapter
InformationRequester : InfoRequester:Specific Adapter
Initial setup - done just once
Initial setup - just done once
Operational phase
1 : Install OAIS-IF adapter()
2 : Install adapter()
3 : Initiate adapter()
4 : Initiate adapter()
5 : registerAdapter()
6 : registerAdapter()
7 : Comm with InformationSource()
8 : Request for Comm()
9 : Comm setup()
10 : getReceiver()
11 : requestPackage()
12 : Component := getComponents()
13 : sendPackage
14 : isInfoUsable()
15 : checkUsability()
16 *[Iterate on options] : agreeStrategy()
17 : usability()
18 : checkOptions()
19 *[Loop until Info is understandable] : requestRepInfoIDs()
20 : checkRIN()
21 : sendRepInfoPackage
22 *[Iterate through RIN] : sendRINInfo
23 : requestRepInfo()
24 : sendRepInfoPackage()
25 : RepInfoID[] := provideRepInfoIDs

Figure 8-2 Sequence diagram

(See here for diagram)

The details of the sequence are as follows:

Activity	**Arrow number**
1) Each will need to buy/borrow/create their specific portion of the adapter that depends upon its specific software design and capabilities e.g.	
a. for an InfoSource - is it a simple file server which we can supplement with a database of RepInfo etc	[1]
b. for an InfoRequester – can it deal with tables and/or images or is it only for displaying documents etc	[2]
2) They each then install the generic portion of the adapter – I think this can be common to all.	[3], [4]
This generic portion does the following things:	
a. Find out about InfoSource using interface we define	
b. Register itself and the InfoSource with the Switchboard e.g. how others can communicate with it – HTTP/port80 or HTTP/port 2678 or REST or CORBA or RMI etc etc	
This is done just once for each source (or requester)	[5], [6]
3) During operations the requester discovers somehow that it needs some information from an InfoSource – how this is done is not covered here. I think lots of work has been done by others, and OAIS does not really provide any guidance.	
4) InfoRequester generic adapter finds out from the Switchboard how to talk to the InfoSource in order to fit into the “OAIS-IF world”, and sets up the communication	[7], [8], [9]
5) The specific adapter of InfoRequester passes on the request to the generic adapter which passes on the request to the InformationSource generic adapter	[10], [11]
6) The generic adapter of the InfoSource receives the request and passes it to the specific adapter which gets the DataObject and RepInfo (at least the start of the RepInfoNetwork) and passes that back	[12]
7) The InfoSource generic adapter creates an PackagedInfo object and sends that to the generic adapter of the InfoRequester	[13]
8) The generic adapter of the InfoRequester asks the specific adapter if this is OK. If not then the generic adapter exchanges messages with the InfoSource generic adapter, which may need to talk to the InfoSource specific adapter	[14], [15], [16], [17], [18]
9) PackagedInformation packages are send containing RepInfo until the InfoRequester is satisfied, or else a strategy	[19], [20], [21],

is decided to use Transformation, if the InfoSource can do that	22]
10) It may be that a Registry of RepInfo (essentially another InformationSource) is available and known to the generic adapter, and additional RepInfo may be obtained from that.	[23], [24]

Additional clarifications:

- The Sequence diagram shows the InformationRequester asks for some information for which (by some mechanism not covered here) it has an identifier.
- On receipt of this request the InformationSender creates an InformationPackage, and then creates a PackagedInformation object by combining that with its PackagingInformation.
- On receipt of the PackagedInformation the Requester decides whether or not it has enough RepInfo to understand/use the data.
- If it has enough RepInfo then there is no need for further requests.
- If on the other hand there is not enough RepInfo then
    - The Sender and Receiver agree on a Strategy.
    - The list of identifiers of RepInfo which the Requester “understands” is sent to the Sender.
    - The Sender then sends a PackagedInformation object which contains the appropriate RepInfo.

8.3 STEVE’S CLASS DIAGRAM

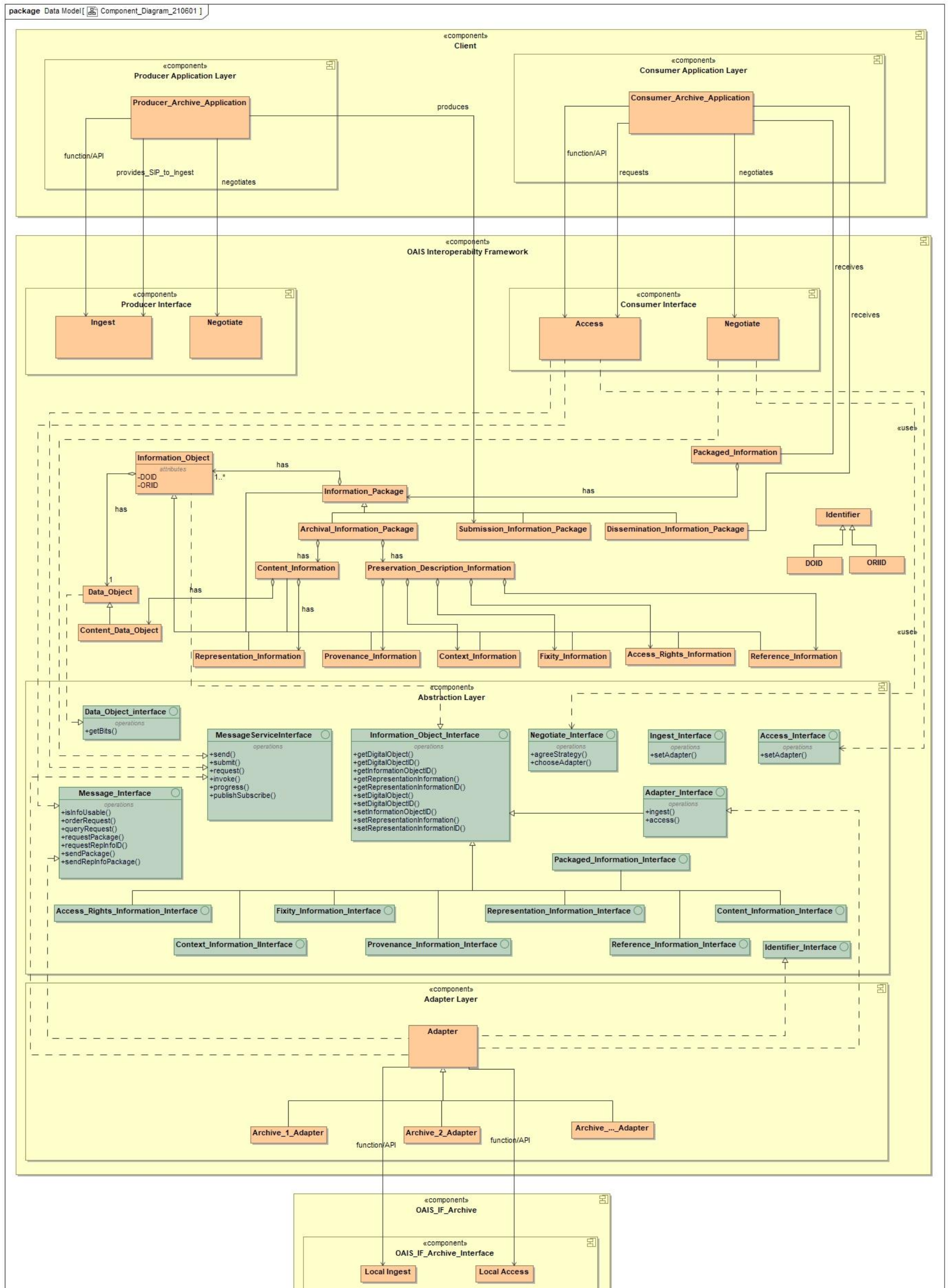
package Data Model[Component_Diagram_210601]
«component»
Client
«component»
Producer Application Layer
Producer_Archive_Application
function/API
provides_SIP_to_Ingest
negotiates
produces
«component»
Consumer Application Layer
Consumer_Archive_Application
function/API
requests
negotiates
receives
receives
«component»
OAIS Interoperability Framework
«component»
Producer Interface
Ingest
Negotiate
«component»
Consumer Interface
Access
Negotiate
«use»
Packaged_Information
Information_Object
attributes
-DOID
-ORIID
1..*
has
Information_Package
has
has
Archival_Information_Package
Submission_Information_Package
Dissemination_Information_Package
Identifier
DOID
ORIID
has
has
Content_Information
Preservation_Description_Information
1
Data_Object
has
has
Content_Data_Object
Representation_Information
Provenance_Information
Context_Information
Fixity_Information
Access_Rights_Information
Reference_Information
«use»
«component»
Abstraction Layer
Data_Object_interface
operations
+getBits()
MessageServiceInterface
operations
+send()
+submit()
+request()
+invoke()
+progress()
+publishSubscribe()
Information_Object_Interface
operations
+getDigitalObject()
+getDigitalObjectID()
+getInformationObjectID()
+getRepresentationInformation()
+getRepresentationInformationID()
+setDigitalObject()
+setDigitalObjectID()
+setInformationObjectID()
+setRepresentationInformation()
+setRepresentationInformationID()
Negotiate_Interface
operations
+agreeStrategy()
+chooseAdapter()
Ingest_Interface
operations
+setAdapter()
Access_Interface
operations
+setAdapter()
Message_Interface
operations
+isInfoUsable()
+orderRequest()
+queryRequest()
+requestPackage()
+requestRepInfoID()
+sendPackage()
+sendRepInfoPackage()
Adapter_Interface
operations
+ingest()
+access()
Packaged_Information_Interface
Access_Rights_Information_Interface
Fixity_Information_Interface
Representation_Information_Interface
Content_Information_Interface
Context_Information_IInterface
Provenance_Information_Interface
Reference_Information_Interface
Identifier_Interface
«component»
Adapter Layer
Adapter
Archive_1_Adapter
Archive_2_Adapter
Archive_..._Adapter
function/API
function/API
«component»
OAIS_IF_Archive
«component»
OAIS_IF_Archive_Interface
Local Ingest
Local Access

Figure 8-3 Steve's composite diagram

8.4 ALTERNATIVE SET OF INTERFACES

FOR UPDATED DIAGRAMS AND DOCUMENTATION SEE
http://www.oais.info/oais-if/javadocs/index.html

In order to clearly separate the OAIS Information Model – which may be used elsewhere, the following diagrams separate the interfaces as follows.

```
└───info
    └───oais
        └───interfaces
            ├───infomodel
            │   ├───identifiers
            │   ├───poss1AccessRights      possible interfaces for Access Rights
            │   ├───poss1Fixity            possible interfaces for Fixity
            │   ├───poss1Provenance        possible interfaces for Provenance
            │   ├───poss1Reference         possible interfaces for Reference
            │   └───poss1RepInfo           possible flag interfaces for RepInfo
            │       └───possRepInfoNetwork  possible interfaces for navigating RIN
            └───oaisif                     OAIS-IF specific interfaces
```

NOTE: any of the objects defined in the OAIS Information Model may be requested. In general one would expect to receive either the object itself or an Identifier which allows the object to be obtained at a later time. The interfaces here are in terms of the objects themselves but these should be expanded to allow the option of just receiving identifiers, for example in the case that the objects are large.

ALL JAVADOCS ARE AVAILABLE AT http://www.oais.info/oais-if/javadocs/index.html

8.4.1 INTERFACE HIERARCHY

NOTE: Links will not work

- info.oais.interfaces.infomodel.poss1RepInfo.possRepInfoNetwork.AcyclicDirectedGraph
    - info.oais.interfaces.infomodel.RepresentationInformationNetwork (also extends info.oais.interfaces.infomodel.RepresentationInformation)
- info.oais.interfaces.oaisif.Adapter
    - info.oais.interfaces.oaisif.GenericAdapter
    - info.oais.interfaces.oaisif.SpecificAdapter
- info.oais.interfaces.oaisif.CommunicationGToG
- info.oais.interfaces.infomodel.DataObject
    - info.oais.interfaces.infomodel.DigitalObject
    - info.oais.interfaces.infomodel.InformationPackage
        - info.oais.interfaces.infomodel.ArchivalInformationPackage
            - info.oais.interfaces.infomodel.ArchivalInformationCollection
            - info.oais.interfaces.infomodel.ArchivalInformationUnit
        - info.oais.interfaces.infomodel.DisseminationInformationPackage
        - info.oais.interfaces.infomodel.SubmissionInformationPackage
    - info.oais.interfaces.infomodel.PhysicalObject
- info.oais.interfaces.infomodel.identifiers.DigitalObjectId
- info.oais.interfaces.infomodel.poss1RepInfo.possRepInfoNetwork.DirectedEdge
- info.oais.interfaces.infomodel.poss1Fixity.FixityEntry
- info.oais.interfaces.oaisif.Generic2Registry
- info.oais.interfaces.oaisif.Generic2Switchboard
- info.oais.interfaces.oaisif.GenericToGenericAdapter
- info.oais.interfaces.oaisif.GenericToSpecificAdapter
    - info.oais.interfaces.oaisif.SpecificSinkToGeneric
    - info.oais.interfaces.oaisif.SpecificSourceToGeneric
- info.oais.interfaces.infomodel.Identifier
    - info.oais.interfaces.infomodel.PersistentIdentifier

        - info.oais.interfaces.infomodel.UniquePersistentIdentifier
    - info.oais.interfaces.infomodel.identifiers.RepInfoId
- info.oais.interfaces.oaisif.InfoEndPointId
- info.oais.interfaces.infomodel.identifiers.InfoObjectId
- info.oais.interfaces.infomodel.InformationObject
    - info.oais.interfaces.infomodel.AccessRightsInformation
        - info.oais.interfaces.infomodel.poss1AccessRights.AccessRightsInt1
    - info.oais.interfaces.infomodel.poss1Provenance.Actor
    - info.oais.interfaces.infomodel.ContentInformation
    - info.oais.interfaces.infomodel.ContextInformation
    - info.oais.interfaces.infomodel.DescriptiveInformation
    - info.oais.interfaces.infomodel.poss1Fixity.FixityEncoding
    - info.oais.interfaces.infomodel.FixityInformation
        - info.oais.interfaces.infomodel.poss1Fixity.FixityInt1
    - info.oais.interfaces.infomodel.poss1Fixity.FixityText
    - info.oais.interfaces.infomodel.PackageDescription
    - info.oais.interfaces.oaisif.PackagedInformation
    - info.oais.interfaces.infomodel.PackagingInformation
    - info.oais.interfaces.infomodel.PreservationDescriptionInformation
    - info.oais.interfaces.infomodel.ProvenanceInformation
        - info.oais.interfaces.infomodel.poss1Provenance.ProvenanceSteps
    - info.oais.interfaces.infomodel.poss1Provenance.ProvenanceStep
    - info.oais.interfaces.infomodel.ReferenceInformation
    - info.oais.interfaces.infomodel.RepresentationInformation
        - info.oais.interfaces.infomodel.OtherRepInfo
        - info.oais.interfaces.infomodel.RepresentationInformationNetwork (also extends info.oais.interfaces.infomodel.poss1RepInfo.possRepInfoNetwork.AcyclicDirectedGraph)
        - info.oais.interfaces.infomodel.SemanticRepInfo
        - info.oais.interfaces.infomodel.StructureRepInfo
    - info.oais.interfaces.infomodel.poss1Provenance.Step
    - info.oais.interfaces.infomodel.poss1Provenance.TimeStamp
- info.oais.interfaces.infomodel.identifiers.Location

- info.oais.interfaces.infomodel.poss1Reference.ReferenceInt1
- info.oais.interfaces.infomodel.poss1RepInfo.RepInfoCategory
- info.oais.interfaces.infomodel.poss1RepInfo.RepInfoGroup
  - info.oais.interfaces.infomodel.poss1RepInfo.AndRepInfoGroup
  - info.oais.interfaces.infomodel.poss1RepInfo.OrRepInfoGroup
  - info.oais.interfaces.infomodel.poss1RepInfo.RepInfoLabel
- info.oais.interfaces.infomodel.poss1AccessRights.RightsEncoding
- info.oais.interfaces.infomodel.poss1AccessRights.RightsText
- info.oais.interfaces.infomodel.poss1RepInfo.possRepInfoNetwork.RootVertex
  - info.oais.interfaces.infomodel.poss1RepInfo.possRepInfoNetwork.Vertex

The following diagrams illustrate the interfaces in each.

8.4.2 INFO.OAIS.INTERFACES.INFOMODEL

This is the full OAIS Information Model – OAISv3.

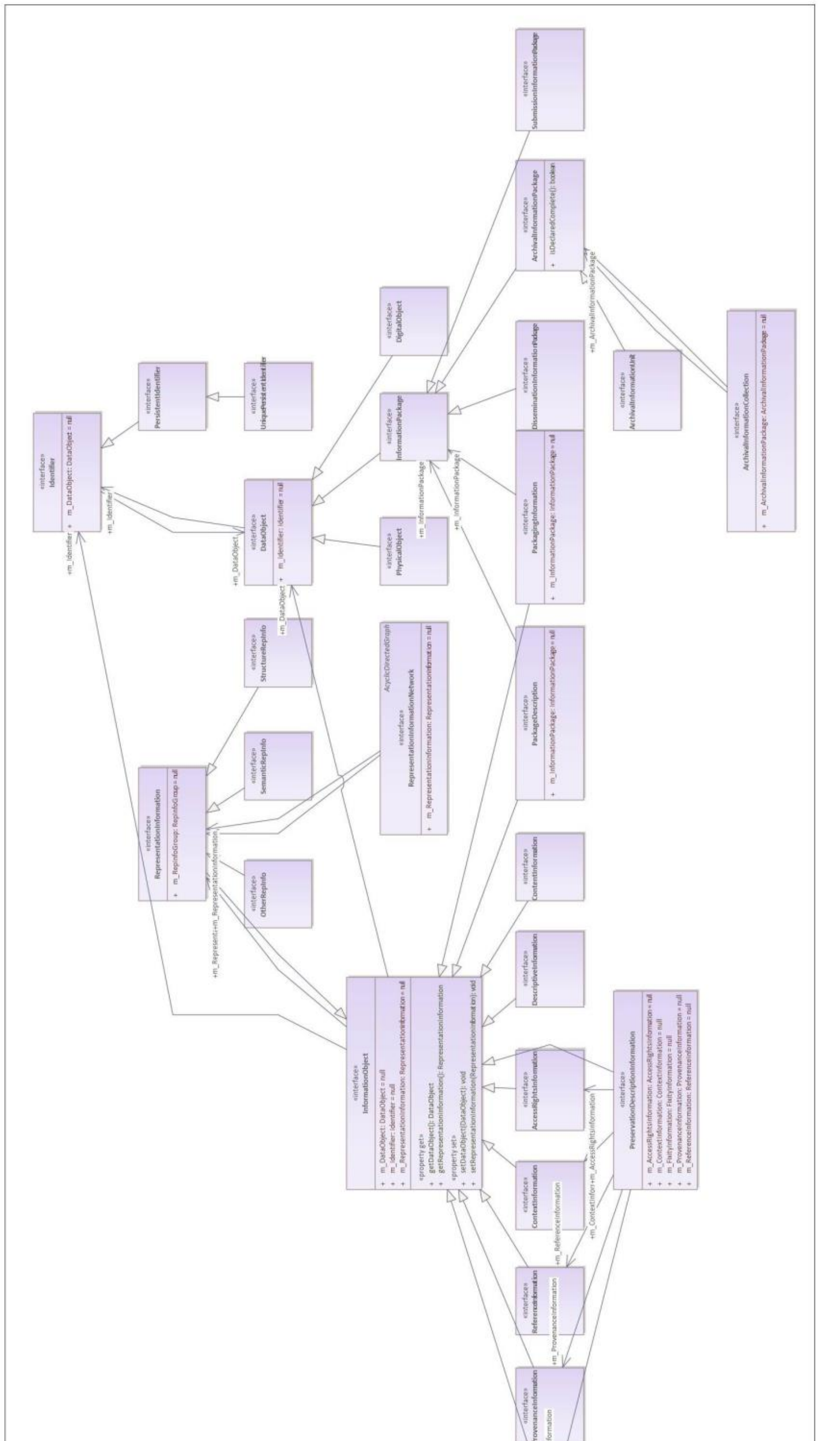
«interface» SubmissionInformationPackage
«interface» ArchivalInformationPackage
+ isDeclaredComplete(): boolean
«interface» DigitalObject
«interface» DisseminationInformationPackage
+m_ArchivalInformationPackage
«interface» ArchivalInformationUnit
«interface» ArchivalInformationCollection
+ m_ArchivalInformationPackage: ArchivalInformationPackage = null
«interface» PersistentIdentifier
«interface» UniquePersistentIdentifier
«interface» Identifier
+ m_DataObject: DataObject = null
«interface» InformationPackage
«interface» DataObject
+ m_Identifier: Identifier = null
«interface» PackagingInformation
+ m_InformationPackage: InformationPackage = null
+m_Identifier
+m_DataObject
+m_InformationPackage
«interface» PhysicalObject
«interface» StructureRepInfo
«interface» SemanticRepInfo
AcyclicDirectedGraph
«interface» RepresentationInformationNetwork
+ m_RepresentationInformation: RepresentationInformation = null
«interface» PackageDescription
+ m_InformationPackage: InformationPackage = null
«interface» RepresentationInformation
+ m_RepInfoGroup: RepInfoGroup = null
«interface» OtherRepInfo
+m_Represents
+m_RepresentationInformation
«interface» ContentInformation
«interface» DescriptiveInformation
«interface» InformationObject
+ m_DataObject: DataObject = null
+ m_Identifier: Identifier = null
+ m_RepresentationInformation: RepresentationInformation = null
«property get»
+ getDataObject(): DataObject
+ getRepresentationInformation(): RepresentationInformation
«property set»
+ setDataObject(DataObject): void
+ setRepresentationInformation(RepresentationInformation): void
«interface» AccessRightsInformation
«interface» PreservationDescriptionInformation
+ m_AccessRightsInformation: AccessRightsInformation = null
+ m_ContextInformation: ContextInformation = null
+ m_FixityInformation: FixityInformation = null
+ m_ProvenanceInformation: ProvenanceInformation = null
+ m_ReferenceInformation: ReferenceInformation = null
+m_AccessRightsInformation
+m_ContextInfo
«interface» ContextInformation
+m_ReferenceInformation
«interface» ReferenceInformation
+m_ProvenanceInformation
«interface» ProvenanceInformation

Figure 8-4 OAIS InfoModel Interfaces

Definitions taken from OAIS version 3.

8.4.2.1 Interface AccessRightsInformation extends InformationObject

The information that identifies the access restrictions pertaining to the Content Data Object, including the legal framework, licensing terms, and access control. It contains the access and distribution conditions stated within the Submission Agreement, related to both preservation (by the OAIS) and final usage (by the Consumer). It also includes the specifications for the application of rights enforcement measures. [OAIS]

8.4.2.1.1 Method Summary

Methods inherited from interface info.oais.interfaces.infomodel.InformationObject

getDataObject, getRepresentationInformation, setDataObject, setRepresentationInformation

8.4.2.2 Interface ArchivalInformationCollection extends ArchivalInformationPackage, DataObject, InformationPackage

An Archival Information Package whose Content Information is an aggregation of other Archival Information Packages.; its PDI must include a description of the collection criteria and process. NOTE - At a minimum all OAISes can be viewed as having at least one AIC which contains all the AIPs held by the OAIS. [OAIS]

8.4.2.2.1 Method Summary

Methods inherited from interface info.oais.interfaces.infomodel.ArchivalInformationPackage

isDeclaredComplete

8.4.2.3 Interface ArchivalInformationPackage extends DataObject, InformationPackage

8.4.2.3.1 All Known Subinterfaces:

ArchivalInformationCollection, ArchivalInformationUnit

An Information Package, consisting of the Content Information and the associated Preservation Description Information (PDI), which is preserved within an OAIS. [OAIS]

8.4.2.3.2 Method Summary

Modifier and Type	Method	Description
boolean	**isDeclaredComplete()**	If an archive declares that the AIP is complete i.e.

8.4.2.3.3 Method Detail

boolean isDeclaredComplete()

If an archive declares that the AIP is complete i.e. is suitable for long term preservation in that archive, then this method should return TRUE.

8.4.2.4 Interface ArchivalInformationUnit extends ArchivalInformationPackage, DataObject, InformationPackage

An Archival Information Package where the Content Information does not include any other Archival Information Packages. [OAIS]

8.4.2.4.1 Method Summary

Methods inherited from interface info.oais.interfaces.infomodel.ArchivalInformationPackage

isDeclaredComplete

8.4.2.5 Interface ContentInformation extends public interface ContentInformation extends InformationObject

A set of information that is the original target of preservation. It is an Information Object composed of its Content Data Object and its Representation Information. [OAIS]

8.4.2.5.1 Method Summary

Methods inherited from interface info.oais.interfaces.infomodel.InformationObject

getDataObject, getRepresentationInformation, setDataObject, setRepresentationInformation

8.4.2.6 Interface ContextInformation extends InformationObject

The information that documents the relationships of the Content Data Object to its environment. This includes why the Content Data Object was created and how it relates to other Content Data Objects. [OAIS]

8.4.2.6.1 Method Summary

Methods inherited from interface info.oais.interfaces.infomodel.InformationObject

getDataObject, getRepresentationInformation, setDataObject, setRepresentationInformation

8.4.2.7 Interface DataObject

8.4.2.7.1 All Known Subinterfaces:

ArchivalInformationCollection, ArchivalInformationPackage, ArchivalInformationUnit, DigitalObject, DisseminationInformationPackage, InformationPackage, PhysicalObject, SubmissionInformationPackage

Either a Physical Object or a Digital Object. [OAIS] Data: A reinterpretable representation of information in a formalized manner suitable for communication, interpretation, or processing. NOTE - Examples of data include a sequence of bits, a table of numbers, the characters on a page, the recording of sounds made by a person speaking, or a moon rock specimen. [OAIS]

8.4.2.8 Interface DescriptiveInformation extends InformationObject

An Information Object which is a set of information, consisting primarily of Package Descriptions, which is provided to Data Management to support the finding, ordering, and retrieving of OAIS information holdings by Consumers.

8.4.2.8.1 Method Summary

Methods inherited from interface info.oais.interfaces.infomodel.InformationObject

getDataObject, getRepresentationInformation, setDataObject, setRepresentationInformation

8.4.2.9 Interface DigitalObject extends DataObject

An object composed of a set of bit sequences. [OAIS]

8.4.2.10 Interface DisseminationInformationPackage extends DataObject, InformationPackage

An Information Package, derived from one or more AIPs, and sent by Archives to the Consumer in response to a request to the OAIS. [OAIS]

8.4.2.11 Interface FixityInformation extends InformationObject

The information which documents the mechanisms that ensure that the Content Data Object has not been altered in an undocumented manner. [OAIS]

8.4.2.11.1 Method Summary

Methods inherited from interface info.oais.interfaces.infomodel.InformationObject

getDataObject, getRepresentationInformation, setDataObject, setRepresentationInformation

8.4.2.12 Interface Identifier

8.4.2.12.1 All Known Subinterfaces:

PersistentIdentifier, UniquePersistentIdentifier

An identifier is a name that identifies (that is, labels the identity of) either a unique object or a unique class of objects, where the "object" or class may be an idea, physical countable object (or class thereof), or physical noncountable substance (or class thereof). [OAIS]

8.4.2.13 Interface InformationObject

8.4.2.13.1 All Known Subinterfaces:

AccessRightsInformation, AndRepInfoGroup, ContentInformation, ContextInformation, DescriptiveInformation, FixityInformation, OrRepInfoGroup, OtherRepInfo, PackageDescription, PackagedInformation, PackagingInformation, PreservationDescriptionInformation, ProvenanceInformation, ReferenceInformation, RepInfoGroup, RepInfoLabel, RepresentationInformation, RepresentationInformationNetwork, SemanticRepInfo, StructureRepInfo

A Data Object together with its Representation Information [OAIS] Information: Any type of knowledge that can be exchanged. In an exchange, it is represented by data.
NOTE - An example of Information is a string of bits (the data) accompanied by a description of how to interpret the string of bits as numbers representing temperature observations measured in degrees Celsius (the Representation Information). [OAIS]

8.4.2.13.2 Method Summary

Modifier and Type	Method	Description
DataObject	**getDataObject ()**	
RepresentationInformation	**getRepresentationInformation ()**	
void	**setDataObject (DataObject dataObj)**	
void	**setRepresentationInformation (RepresentationInformation repInfo)**	

8.4.2.13.3 Method Detail

DataObject getDataObject()

RepresentationInformation getRepresentationInformation()

void setDataObject(DataObject dataObj)

Parameters:

dataObj - The

void setRepresentationInformation(RepresentationInformation repInfo)

Parameters:

repInfo - repInfo

8.4.2.14 Interface InformationPackage extends DataObject

8.4.2.14.1 All Known Subinterfaces:

ArchivalInformationCollection, ArchivalInformationPackage, ArchivalInformationUnit, DisseminationInformationPackage, SubmissionInformationPackage

A logical container composed of optional Information Object(s). Associated with this Information Package is Packaging Information used to delimit and identify the Information Object and optional Package Description information used to facilitate searches for the Information Object. [OAIS]

8.4.2.15 Interface OtherRepInfo extends InformationObject, RepresentationInformation

A type of Representation Information which cannot easily be classified as Structure Representation Information or Semantic Representation Information. It is a type of Information Object. NOTE - For example, software, algorithms, encryption, written instructions and many other things may be needed to understand the Content Data Object in ways exemplified by the Preservation Objectives, all of which therefore would be, by definition, Representation Information, yet would not obviously be either Structure Representation Information or Semantic Representation. Information defining how the Structure Representation Information and the Semantic Representation Information relate to each other, or software needed to process a database file would also be regarded as Other Representation Information. [OAIS]

8.4.2.15.1 Method Summary

Methods inherited from interface info.oais.interfaces.infomodel.InformationObject

getDataObject, getRepresentationInformation, setDataObject, setRepresentationInformation

8.4.2.16 Interface PackageDescription extends InformationObject

The information intended for use by Access Aids. It is a type of Information Object. [OAIS]

8.4.2.16.1 Method Summary

Methods inherited from interface info.oais.interfaces.infomodel.InformationObject

getDataObject, getRepresentationInformation, setDataObject, setRepresentationInformation

8.4.2.17 Interface PackagingInformation extends InformationObject

The information that describes how the components of an Information Package are logically or physically bound together and how to identify and extract the components. It is a type of Information Object. [OAIS]

8.4.2.17.1 Method Summary

Methods inherited from interface info.oais.interfaces.infomodel.InformationObject

getDataObject, getRepresentationInformation, setDataObject, setRepresentationInformation

8.4.2.18 Interface PersistentIdentifier extends Identifier

8.4.2.18.1 All Known Subinterfaces:

UniquePersistentIdentifier

A long-lasting Identifier. NOTE - Persistent Identifier resolution services cannot be guaranteed to persist. The Archive should evaluate what the options are if the resolver service ceases to operate. The archive may also consider using multiple alternative Persistent Identifier systems for an object. [OAIS]

8.4.2.19 Interface PhysicalObject extends DataObject

An object (such as a moon rock, bio-specimen, microscope slide) with physically observable properties that represent information that is considered suitable for being adequately documented for preservation, distribution, and independent usage. [OAIS]

8.4.2.20 Interface PreservationDescriptionInformation extends InformationObject

The information, which along with Representation Information, is necessary for adequate preservation of the Content Data Object and which can be categorized as Provenance Information, Context Information, Reference Information, Fixity Information, and Access Rights Information. It is a type of Information Object. NOTE Defining PDI (as well as its components: Provenance Information, Context Information, Reference Information, Fixity Information, and Access Rights Information) as relevant to the Content Data Object does not mean that those concerns are any less important for other data objects or at other levels, for example, it is important to apply reference, fixity, provenance, context and access rights to Representation Information, or to any other information the Archive is preserving. Definition of these terms as relevant to the Content Data Object is simply to ease discussion of these concepts at the Content Data Object level. [OAIS]

8.4.2.20.1 Method Summary

Methods inherited from interface info.oais.interfaces.infomodel.InformationObject

getDataObject, getRepresentationInformation, setDataObject, setRepresentationInformation

8.4.2.21 Interface ProvenanceInformation extends InformationObject

The information that documents the history of the Content Data Object. This information tells the origin or source of the Content Data Object, any changes that may have taken place since it was originated, and who has had custody of it since it was originated. The Archive is responsible for creating and preserving Provenance Information from the point of Ingest; however, earlier Provenance Information should be provided by the Producer. Provenance Information adds to the evidence to support Authenticity. [OAIS]

8.4.2.21.1 Method Summary

Methods inherited from interface info.oais.interfaces.infomodel.InformationObject

getDataObject, getRepresentationInformation, setDataObject, setRepresentationInformation

8.4.2.22 Interface ReferenceInformation extends InformationObject

The information that is used as an identifier for the Content Data Object. It also includes identifiers that allow outside systems to refer unambiguously to a particular Content Data Object. NOTE - An example of Reference Information is an ISBN. [OAIS]

8.4.2.22.1 Method Summary

Methods inherited from interface info.oais.interfaces.infomodel.InformationObject

getDataObject, getRepresentationInformation, setDataObject, setRepresentationInformation

8.4.2.23 Interface RepresentationInformation extends InformationObject

8.4.2.23.1.1 All Known Subinterfaces:

AndRepInfoGroup, OrRepInfoGroup, OtherRepInfo, RepInfoGroup, RepInfoLabel, RepresentationInformationNetwork, SemanticRepInfo, StructureRepInfo

The information that maps a Data Object into more meaningful concepts so that the Data Object may be understood in ways exemplified by Preservation Objectives. It is a type of Information Object. NOTE - An example of Representation Information for a bit sequence which is a FITS file might consist of the FITS standard which defines the format plus a dictionary which defines the meaning in the file of keywords which are not part of the standard. This would then allow the information in the FITS file to be used by a computer program to display the image which may be contained in the FITS file, together with the associated coordinate system so that a human can identify objects of interest, for example stars or galaxies. Alternatively, the computer program may identify such objects automatically.

8.4.2.23.2 Method Summary

Methods inherited from interface info.oais.interfaces.infomodel.InformationObject

getDataObject, getRepresentationInformation, setDataObject, setRepresentationInformation

8.4.2.24 Interface RepresentationInformationNetwork extends AcyclicDirectedGraph, InformationObject, RepresentationInformation

The set of Representation Information that fully describes the meaning of a Data Object. Representation Information in digital forms needs additional Representation Information so its digital forms can be understood over the Long Term. It is a type of Information Object. [OAIS]

8.4.2.24.1 Method Summary

Methods inherited from interface info.oais.interfaces.infomodel.InformationObject

getDataObject, getRepresentationInformation, setDataObject, setRepresentationInformation

8.4.2.25 Interface SemanticRepInfo extends InformationObject, RepresentationInformation

The Representation Information that further describes the meaning of the Data Object, and its parts or elements, beyond that provided by the Structure Representation Information. NOTE - For example, Semantic Representation Information may describe the meaning of columns, and perhaps particular values seen in the columns of a spreadsheet. [OAIS]

8.4.2.25.1 Method Summary

Methods inherited from interface info.oais.interfaces.infomodel.InformationObject getDataObject, getRepresentationInformation, setDataObject, setRepresentationInformation

8.4.2.26 Interface StructureRepInfo extends InformationObject, RepresentationInformation

The Representation Information that imparts information about the arrangement of and the organization of the parts or elements of the Data Object. NOTE - For example, Structure Representation Information maps bit streams to common computer types such as characters, numbers, and pixels and aggregations of those types such as character strings and arrays. [OAIS]

8.4.2.26.1 Method Summary

Methods inherited from interface info.oais.interfaces.infomodel.InformationObject

getDataObject, getRepresentationInformation, setDataObject, setRepresentationInformation

8.4.2.27 Interface SubmissionInformationPackage extends DataObject, InformationPackage

An Information Package that is delivered by the Producer to the OAIS for use in the construction or update of one or more AIPs and/or the associated Descriptive Information. [OAIS]

8.4.2.28 Interface UniquePersistentIdentifier extends Identifier, PersistentIdentifier

A Persistent Identifier which is unique within a specific naming convention. [OAIS]

8.4.3 INFO.OAIS.INTERFACES.INFOMODEL.IDENTIFIERS

These additional identifiers re etclate to the more general Identifier interface. They may be used as “flag” interfaces e.g. to make it clear that this Identifier is for a DigitalObject

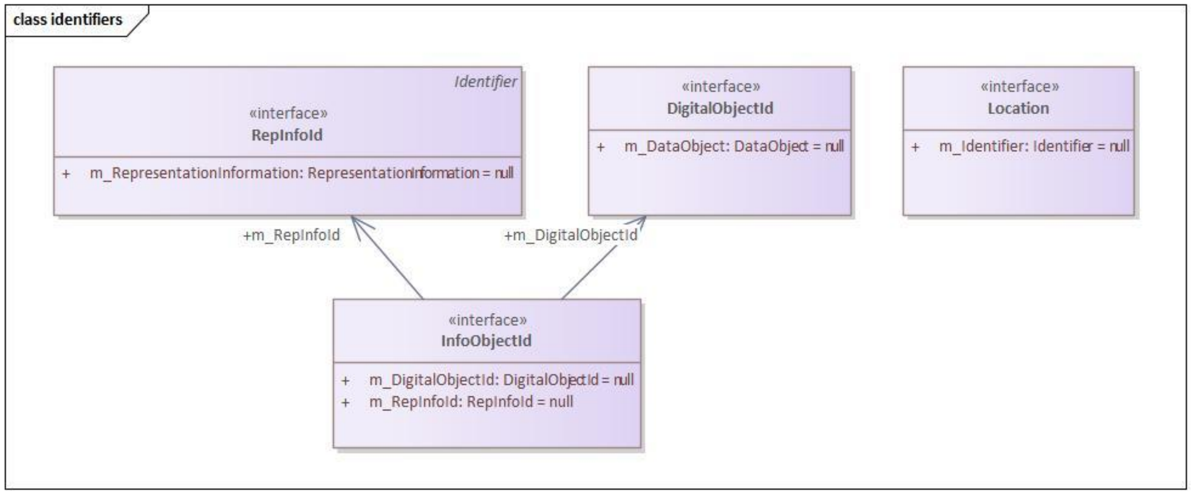
class identifiers
Identifier
«interface»
RepInfoId
+ m_RepresentationInformation: RepresentationInformation = null
«interface»
DigitalObjectId
+ m_DataObject: DataObject = null
«interface»
Location
+ m_Identifier: Identifier = null
+m_RepInfoId
+m_DigitalObjectId
«interface»
InfoObjectId
+ m_DigitalObjectId: DigitalObjectId = null
+ m_RepInfoId: RepInfoId = null

Figure 8-5 Identifier Interfaces

8.4.3.1 Interface DigitalObjectId extends Identifier

An Identifier which points to a DataObject.

8.4.3.2 Interface InfoObjectId extends Identifier

An Identifier which points to an InformationObject.

8.4.3.3 Interface Location extends Identifier

An Identifier which gives the location of an object

8.4.3.4 Interface RepInfoId extends Identifier

An Identifier which points to a RepresentationInformationObject.

8.4.4 INFO.OAIS.INTERFACES.INFOMODEL.REPINFO

The following interfaces are incomplete ideas for navigating a Representation Information Network. The need for these is that one expects that RepInfo may be grouped. For example one might have alternatives e.g. software that is in the form of either Java OR C#

- the choice is up to the user/user software. Also one might have pieces of RepInfo that should be used together e.g. one almost certainly needs the Structure PLUS the Semantic RepInfo.

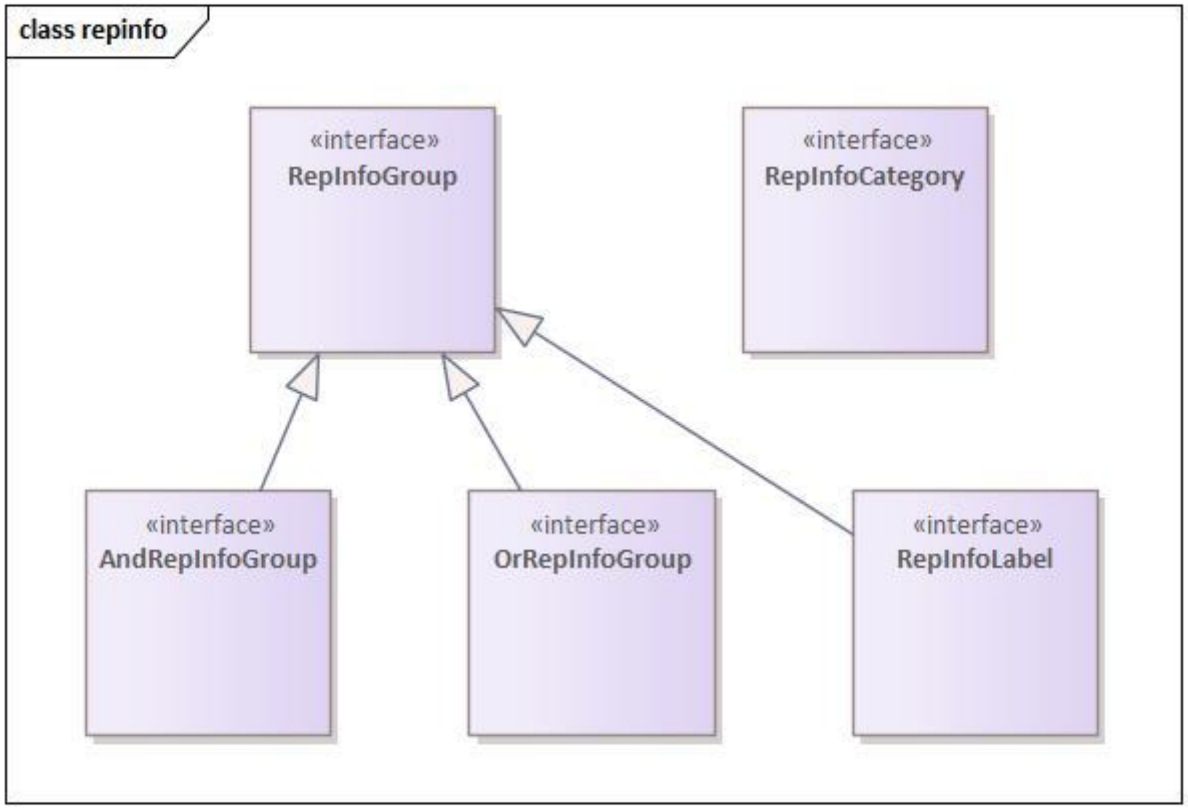

Figure 8-6 RepInfo grouping interfaces

8.4.4.1 Interface AndRepInfoGroup extends InformationObject, RepInfoGroup, RepresentationInformation

All the elements of an AndRepInfoGroup must be used together to be useful.

8.4.4.1.1 Method Summary

Methods inherited from interface info.oais.interfaces.infomodel.InformationObject

getDataObject, getRepresentationInformation, setDataObject, setRepresentationInformation

8.4.4.2 Interface OrRepInfoGroup extends InformationObject, RepInfoGroup, RepresentationInformation

The components of the OrRepInfoGroup should be treated as alternatives i.e. only one of the collection is needed, and the selection will depend upon what other RepInfo is available. For example software may be available as Java code or C# code. If a Java compiler and Virtual machine are available then the Java code should be chosen. If both a Java compiler and a C# compiler are available then either may be chosen.

Method Summary

Methods inherited from interface info.oais.interfaces.infomodel.InformationObject

getDataObject, getRepresentationInformation, setDataObject, setRepresentationInformation

8.4.4.3 Interface RepInfoCategory

A classification of RepInfo [TBD] to facilitate locating the kind of RepInfo wanted.

8.4.4.4 Interface RepInfoGroup extends InformationObject, RepresentationInformation

8.4.4.4.1 All Known Subinterfaces:

AndRepInfoGroup, OrRepInfoGroup, RepInfoLabel

Method Summary

Methods inherited from interface info.oais.interfaces.infomodel.InformationObject

getDataObject, getRepresentationInformation, setDataObject, setRepresentationInformation

8.4.4.5 Interface RepInfoLabel extends InformationObject, RepInfoGroup, RepresentationInformation

Method Summary

Methods inherited from interface info.oais.interfaces.infomodel.InformationObject

getDataObject, getRepresentationInformation, setDataObject, setRepresentationInformation

8.4.5 INFO.OAIS.INTERFACES.INFOMODEL.REPINFO. POSSREPINFONETWORK

A RepInfoNetwork is an Acyclic Directed Graph. There are a number of existing APIs to deal with Directed Graphs, however they are quite complex and so a simple set of interfaces are suggested below.
The existing libraries may be used to implement these interfaces.

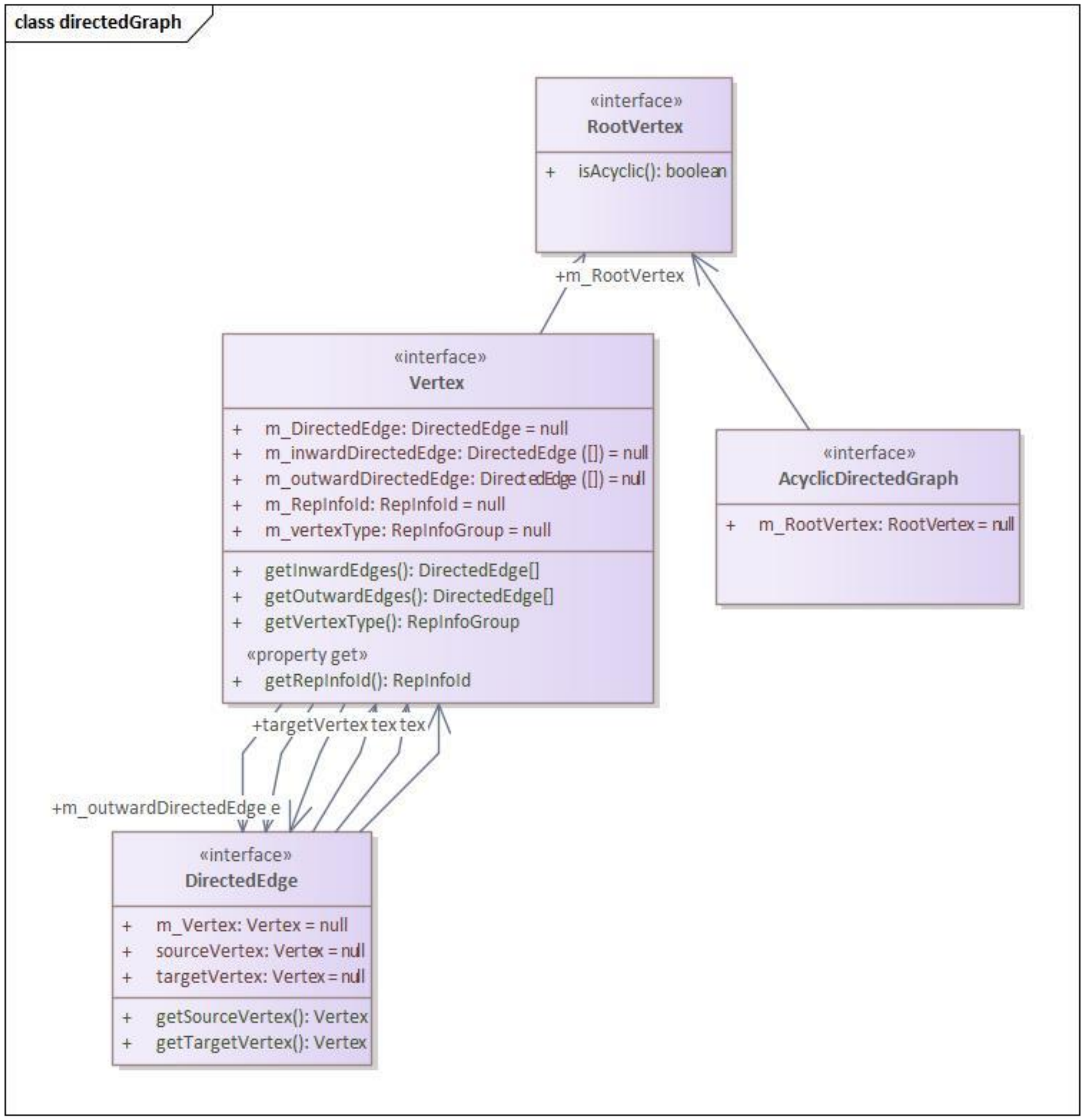
class directedGraph
«interface»
RootVertex
+ isAcyclic(): boolean
+m_RootVertex
«interface»
Vertex
+ m_DirectedEdge: DirectedEdge = null
+ m_inwardDirectedEdge: DirectedEdge ([]) = null
+ m_outwardDirectedEdge: DirectedEdge ([]) = null
+ m_RepInfoId: RepInfoId = null
+ m_vertexType: RepInfoGroup = null
+ getInwardEdges(): DirectedEdge[]
+ getOutwardEdges(): DirectedEdge[]
+ getVertexType(): RepInfoGroup
«property get»
+ getRepInfoId(): RepInfoId
«interface»
AcyclicDirectedGraph
+ m_RootVertex: RootVertex = null
+targetVertex tex tex
+m_outwardDirectedEdge e
«interface»
DirectedEdge
+ m_Vertex: Vertex = null
+ sourceVertex: Vertex = null
+ targetVertex: Vertex = null
+ getSourceVertex(): Vertex
+ getTargetVertex(): Vertex

Figure 8-7 Directed Graph Interfaces

The methods here do not include the set methods.

The aim is to identify Vertexes which have incoming or outgoing directedEdges. The RepInfoNetwork is a Directed Graph of DirectedEgdes and Vertexes.

Each Vertex may have a pointer to an associated piece of RepInfo, for which the outgoing edges provide the RepInfo. The edge may point to a Vertex which may simply be a way to a single piece of RepInfo or a group that may be an “AndGroup” or an “OrGroup”. The former must be taken together, for example Structure plus Semantic plus Other RepInfo, while the latter provides alternatives. Using interfaces derived from these ideas one can navigate the RepInfoNetwork, or serialise it as a serialised Java Object.

8.4.5.1 Interface AcyclicDirectedGraph

8.4.5.1.1 All Known Subinterfaces:

RepresentationInformationNetwork

The acyclic directed graph has one root vertex.

8.4.5.2 Interface DirectedEdge

Each edge has exactly one Source Vertex and one Target Vertex.

8.4.5.2.1 Method Summary

Modifier and Type	Method	Description
Vertex	**getSourceVertex ()**	Returns the Vertex from which the Edge starts.
Vertex	**getTargetVertex ()**	Returns the Vertex at which the Edge ends.

8.4.5.2.2 Method Detail

Vertex getSourceVertex()

Returns the Vertex from which the Edge starts.

Returns: sourceVertex The originating end of the Edge i.e. the Edge goes from the source to the target.

Vertex getTargetVertex()

Returns the Vertex at which the Edge ends.

Returns: tergetVertex The target end of the Edge i.e. the Edge goes from the source to the target.

8.4.5.3 Interface RootVertex

8.4.5.3.1 All Known Subinterfaces:

Vertex

The unique root Vertex of the acyclic directed graph.

8.4.5.3.2 Method Summary

Modifier and Type	Method	Description
boolean	**isAcyclic ()**	This is a method which can be used to check whether or not the graph is acyclic.

8.4.5.3.3 Method Detail

boolean isAcyclic()

This is a method which can be used to check whether or not the graph is acyclic. For example if a Vertex or an Edge is added this can be used to check if this would cause the graph to contain a cycle i.e. FAIL to be acyclic, in which case the added component would be removed.

8.4.5.4 Interface Vertex extends RootVertex

A Vertex points to zero or more other Vertexs (sic), and it is pointed to by at least one other Vertex, unless it is the RootVertex. The VertexType tells one whether the the Vertexs pointed to should be taken together or are alternatives. A Vertex may have an associated RepresentationInformationObject.

8.4.5.4.1 Method Summary

Modifier and Type	Method	Description
DirectedEdge[]	**getInwardEdges ()**	Returns an array of Edges which end at this Vertex
DirectedEdge[]	**getOutwardEdges ()**	Returns an array of Edges which start at this Vertex
RepInfoId	**getRepInfoId ()**	

Methods inherited from interface info.oais.interfaces.infomodel.repinfo.directedGraph.RootVertex
isAcyclic

8.4.5.4.2 Method Detail

DirectedEdge[] getInwardEdges()
Returns an array of Edges which end at this Vertex
Returns: inwardEdges The array of Edges which end at this Vertex
getOutwardEdges
DirectedEdge[] getOutwardEdges()
Returns an array of Edges which start at this Vertex
Returns: outwardEdges The array of Edges which start at this vertex.
RepInfoId getRepInfoId()

8.4.6 INFO.OAIS.INTERFACES.OAISIF

The following interfaces make an INCOMPLETE start at trying to define interfaces for the adapters in a way consistent with the Sequence Diagram.

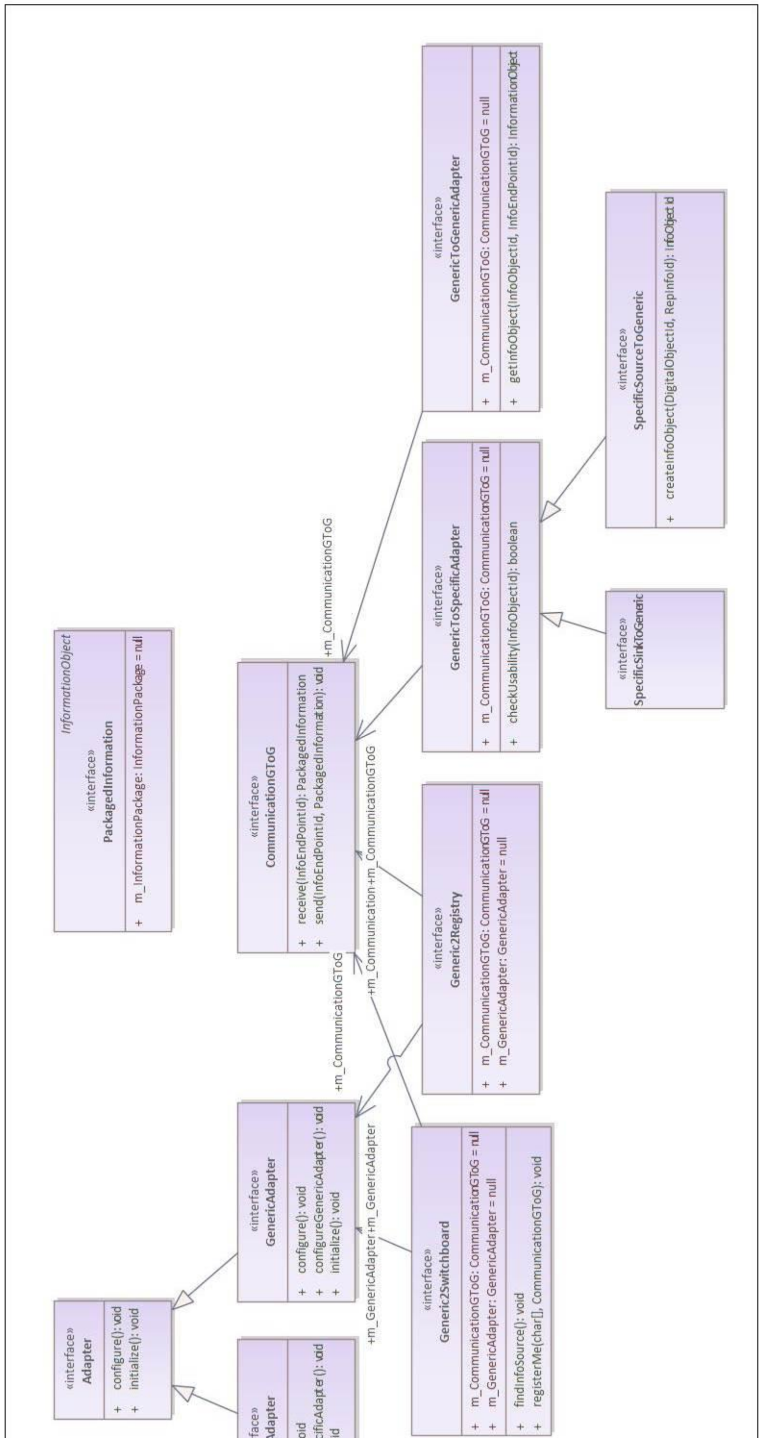
«interface» GenericToGenericAdapter
+ m_CommunicationGToG: CommunicationGToG = null
+ getInfoObject(InfoObjectId, InfoEndPointId): InformationObject
«interface» SpecificSourceToGeneric
+ createInfoObject(DigitalObjectId, RepInfoId): InfoObjectId
«interface» GenericToSpecificAdapter
+ m_CommunicationGToG: CommunicationGToG = null
+ checkUsability(InfoObjectId): boolean
«interface» SpecificSinkToGeneric
InformationObject
«interface» PackagedInformation
+ m_InformationPackage: InformationPackage = null
«interface» CommunicationGToG
+ receive(InfoEndPointId): PackagedInformation
+ send(InfoEndPointId, PackagedInformation): void
+m_CommunicationGToG
+m_CommunicationGToG
+m_CommunicationGToG
+m_Communication
+m_CommunicationGToG
«interface» Generic2Registry
+ m_CommunicationGToG: CommunicationGToG = null
+ m_GenericAdapter: GenericAdapter = null
«interface» GenericAdapter
+ configure(): void
+ configureGenericAdapter(): void
+ initialize(): void
+m_GenericAdapter
+m_GenericAdapter
+m_GenericAdapter
«interface» Generic2Switchboard
+ m_CommunicationGToG: CommunicationGToG = null
+ m_GenericAdapter: GenericAdapter = null
+ findInfoSource(): void
+ registerMe(char[], CommunicationGToG): void
«interface» Adapter
+ configure(): void
+ initialize(): void
«interface»
Adapter
oid
cificAdapter(): void
id

Figure 8-8 OAIS-IF specific interfaces

8.4.6.1 Interface Adapter

8.4.6.1.1 All Known Subinterfaces:

GenericAdapter, SpecificAdapter

8.4.6.1.2 Method Summary

Modifier and Type	Method	Description
void	**configure ()**	To be overridden by Specific and Generic Adapters
void	**initialize ()**	To be overridden by Specific and Generic Adapters

8.4.6.1.3 Method Detail

void configure()

To be overridden by Specific and Generic Adapters

void initialize()

To be overridden by Specific and Generic Adapters

8.4.6.2 Interface CommunicationGToG

Communications interface. It covers communications between - Generic to Generic adapters - Generic to Specific adapters - Generic to Switchboard - Generic to Registry The communications may occur over a Network (Wide Area or Local), or within a computer system e.g. process to process communication, which would be the case where the Generic and the Specific adapters are on the same computer system.

8.4.6.2.1 Method Summary

Modifier and Type	Method	Description

PackagedInformation	**receive** **(InfoEndPointId provider)**	
void	**send** **(InfoEndPointId target, PackagedInformation pkgInfo)**	

8.4.6.2.2 Method Detail

PackagedInformation receive(InfoEndPointId provider)

Parameters:

provider - provider

void send(InfoEndPointId target,
PackagedInformation pkgInfo)

Parameters:

target -

pkgInfo - pkgInfo

8.4.6.2.3 Interface Generic2Registry

The additional methods used to communicate between a Generic Adapter and a Registry. The methods for communicating with any other source of information will also be needed.

8.4.6.3 Interface Generic2Switchboard

The additional methods used to communicate between a Generic Adapter and a Switchboard. The methods for communicating with any other source of information will also be needed. The Generic adapter should be configured to know the name of the service is is being used for (e.g. the URI) and also the communications mechanism to use e.g. HTTP at port 7778.

8.4.6.3.1 Method Summary

Modifier and Type	Method	Description
void	**findInfoSource** **()**	

void	**registerMe (char[] myName, CommunicationGToG myComms)**	

8.4.6.3.2 Method Detail

void findInfoSource()

void registerMe(char[] myName, CommunicationGToG myComms)

Parameters:

myName -

myComms - myComms

8.4.6.4 Interface GenericAdapter extends Adapter

The generic adapter interface

8.4.6.4.1 Method Summary

Modifier and Type	Method	Description
void	**Configure ()**	This method allows the Generic adapter to be configures including - setting the name of its service by which it will be know to the switchboard - setting its communications mechanism e.g.
void	**configureGenericAdapter ()**	This method allows the Generic adapter to be configures including - setting the name of its service by which it will be know to the switchboard - setting its communications mechanism e.g.
void	**Initialize ()**	Initialize the Generic Adapter so that it will be usable.

8.4.6.4.2 Method Detail

void configure()

This method allows the Generic adapter to be configures including - setting the name of its service by which it will be know to the switchboard - setting its communications mechanism e.g. --- HTTP port 778 or --- RMI or --- CORBA or --- WSDL or --- TCP port 7771 - authentication mechanisms accepted e.g. --- local username/password or --- LDAP or --- SAML

Specified by:

configure in interface Adapter

void configureGenericAdapter()

This method allows the Generic adapter to be configures including - setting the name of its service by which it will be know to the switchboard - setting its communications mechanism e.g. --- HTTP port 778 or --- RMI or --- CORBA or --- WSDL or --- TCP port 7771 - authentication mechanisms accepted e.g. --- local username/password or --- LDAP or --- SAML

void initialize()

Initialize the Generic Adapter so that it will be usable. This includes - registering with a Switchboard

Specified by:

initialize in interface Adapter

8.4.6.5 Interface GenericToGenericAdapter

These are the methods used to communicate between two Generic Adapters e.g. one at the user side and one at the source side.

8.4.6.5.1 Method Summary

Modifier and Type	Method	Description
InformationObject	**getInfoObject (InfoObjectId info, InfoEndPointId infoSource)**	

8.4.6.6 Method Detail

InformationObject getInfoObject(InfoObjectId info,
InfoEndPointId infoSource)

Parameters:

info -

infoSource - infoSource

8.4.6.7 Interface GenericToSpecificAdapter

8.4.6.7.1 All Known Subinterfaces:

SpecificSinkToGeneric, SpecificSourceToGeneric

8.4.6.7.2 Method Summary

Modifier and Type	Method	Description
boolean	**checkUsability (InfoObjectId infoCurrent)**	Is the information object usable by the Spcific Adapter? If not then more RepresentationInformation is required.

8.4.6.7.3 Method Detail

boolean checkUsability(InfoObjectId infoCurrent)

Is the information object usable by the Spcific Adapter? If not then more RepresentationInformation is required.

Parameters:

infoCurrent - infoCurrent

8.4.6.8 Interface InfoEndPointId

The identifier for the endpoint of a communication. It may be either a source or a sink.

8.4.6.9 Interface PackagedInformation extends InformationObject

An InformationObject which consists of an InformationPackage (as theDataObject) together with its ReprepresentationInformation.

8.4.6.9.1 Method Summary

Methods inherited from interface info.oais.interfaces.infomodel.InformationObject

getDataObject, getRepresentationInformation, setDataObject, setRepresentationInformation

8.4.6.10 Interface SpecificAdapter extends Adapter

The specific adapter interface used by the generic adapter.

8.4.6.10.1 Method Summary

Modifier and Type	Method	Description
void	**configure ()**	Configure specific adapter e.g.
void	**configureSpecificAdapter ()**	Configure specific adapter e.g.

void	**initialize ()**	Initialize the Specific Adapter so that it will be usable.

8.4.6.10.2 Method Detail

void configure()

Configure specific adapter e.g. - association between data object and the various components of the associated AIP. Only RepresentatioInformation must be present; the others may be NULL.

Specified by:

configure in interface Adapter

void configureSpecificAdapter()

Configure specific adapter e.g. - association between data object and the various components of the associated AIP. Only RepresentatioInformation must be present; the others may be NULL.

void initialize()

Initialize the Specific Adapter so that it will be usable. This includes - interacting with its associated software e.g. software for an archive, or user software.

Specified by:

initialize in interface Adapter

8.4.6.11 Interface SpecificSinkToGeneric extends GenericToSpecificAdapter

This interface includes methods to be used when information is to be received. The SINK is a generalised consumer (i.e. sink) of information during an exchange. A particular component in the system may be a SINK and a SOURCE at different times.

8.4.6.11.1 Method Summary

Methods inherited from interface info.oais.interfaces.oaisif.GenericToSpecificAdapter

checkUsability

8.4.6.12 Interface SpecificSourceToGeneric extends GenericToSpecificAdapter

This interface includes methods to be used when information is to be send. The SOURCE is a generalised supplier of information during an exchange. A particular component in the system may be a SINK and a SOURCE at different times.

8.4.6.12.1 Method Summary

Modifier and Type	Method	Description
void	**createInfoObject (DigitalObjectId data, InformationObject info)**	Given a DataObjectId this method finds the associated RepInfoId in the form of the start of the RepresentationInformationNetwork.

Methods inherited from interface info.oais.interfaces.oaisif.GenericToSpecificAdapter

checkUsability

8.4.6.12.2 Method Detail

void createInfoObject(DigitalObjectId data, InformationObject info)

Given a DataObjectId this method finds the associated RepInfoId in the form of the start of the RepresentationInformationNetwork.

Parameters:

data -

info - info

8.5 COMPONENT DIAGRAM

Component diagrams can be used to model the logical components that make up a system. They can be used to model the applications of an organization including their Provided and Required Interfaces and the information that is exchanged between the interfaces.

The diagram below shows the components shows that any system will have, besides the Native System, a Specific Adapter and a copy of the Generic Adapter. The interfaces are indicated. A “System” represents any of the entities which exchange information using OAIS-IF.

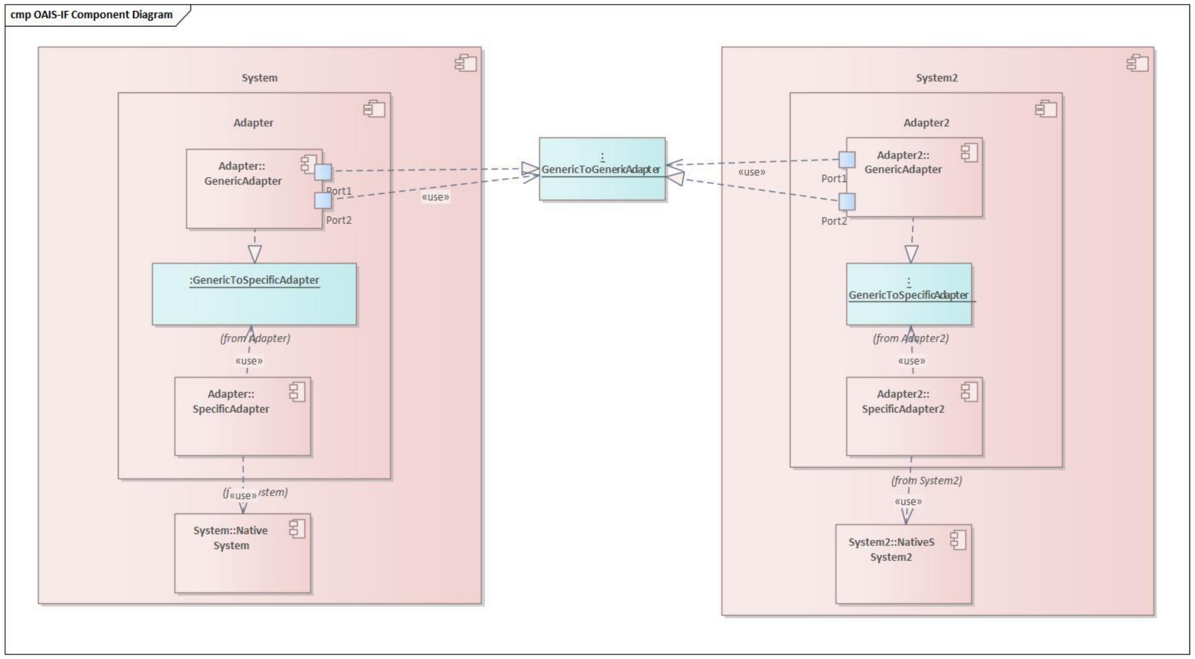

Figure 8-9 Component diagram

There is a negotiation, shown in the Sequence Diagram to decide on whether, in order for the Consumer to be able to use the information returned, the source of information can either:

1. Send additional RepInfo to extend the RepInfo Network so that the Consumer will have enough RepInfo. The Producer may use a Registry/Repository of Representation Information to supplement the RepInfo it already has.
2. The Data Object could be Transformed to match what the Consumer can deal with.

To implement this, CCSDS (or one or more agencies) can provide Reference Implementations. A Domain Archive could initially simply use the Reference Implementation. It would only need to create the Interoperability Table, and supply the Applications (if any) to perform the Transformation, which should be chosen based on maintaining the Transformational Information Properties.

8.6 DEPLOYMENT DIAGRAM

The Deployment diagram indicates (some of the) the software which runs on the physical nodes (e.g. computers).

This shows nodes which match the OAIS-IF scenarios discussed in section 3.

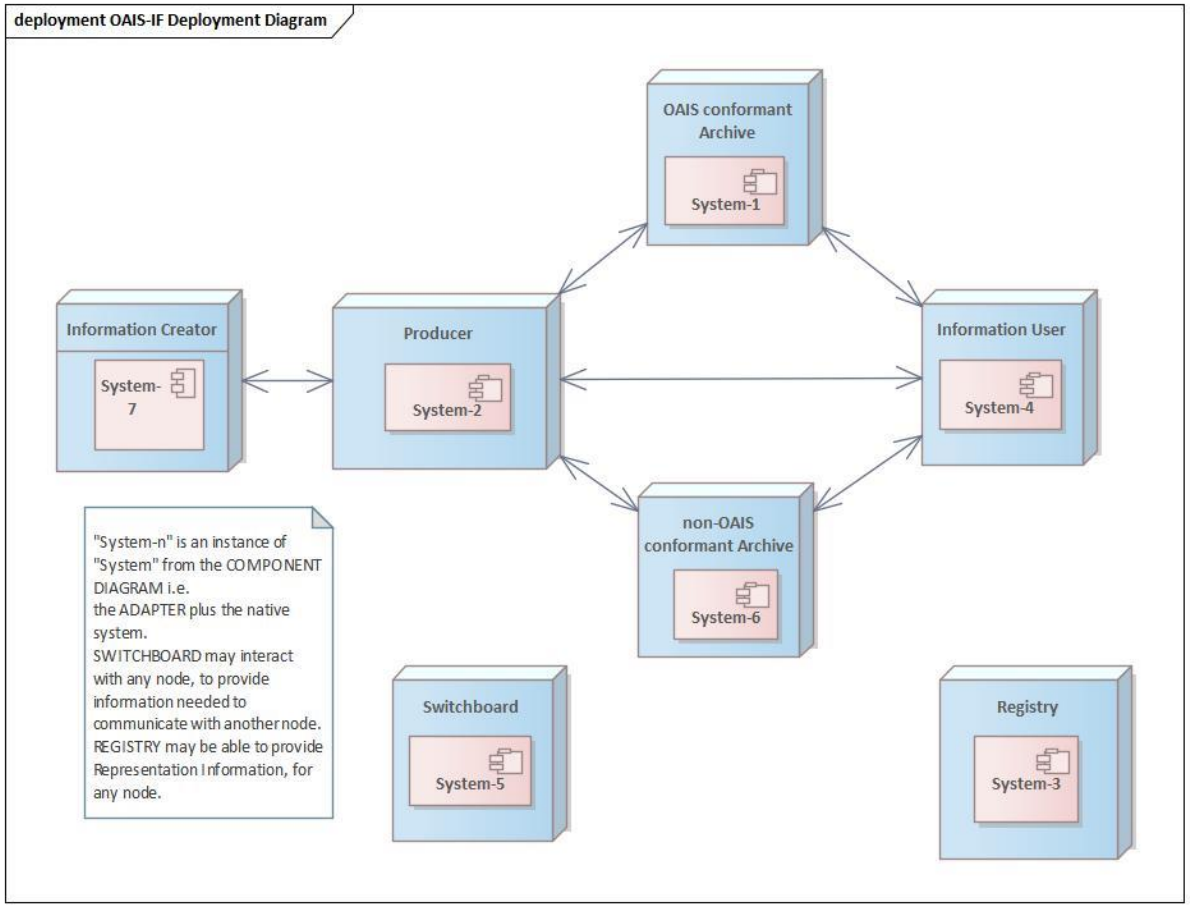
deployment OAIS-IF Deployment Diagram
OAIS conformant Archive
System-1
Information Creator
System-7
Producer
System-2
Information User
System-4
non-OAIS conformant Archive
System-6
"System-n" is an instance of "System" from the COMPONENT DIAGRAM i.e. the ADAPTER plus the native system. SWITCHBOARD may interact with any node, to provide information needed to communicate with another node. REGISTRY may be able to provide Representation Information, for any node.
Switchboard
System-5
Registry
System-3

Figure 8-10 Deployment diagram

8.7 TODO:

- QUERY - could use the Java Query Interface https://docs.oracle.com/javaee/7/api/javax/persistence/Query.html

This is not quite consistent with the CCSDS Message Abstraction Layer (see https://public.ccsds.org/Pubs/521x0b2e1.pdf), but if this seems a good way to go it may be relatively straightforward to do that. We then also may be able to use the MAL JAVA API (see https://public.ccsds.org/Pubs/523x1m1.pdf). A JAVA prototype implementation could be quite straightforward.

The binding to the MAL HTTP Transport (see https://public.ccsds.org/Pubs/524x3b1.pdf) may also be useful.

9

10 ADDITIONAL DETAILED INTERFACES

10.1 PURPOSE

This note, which is an introduction to [3], presents a high level view of an approach to the design of OAIS-IF which will answer this question. If this approach is accepted then we can look at the details of how this could be taken further, perhaps by taking [3] forward.

Therefore, a reviewer of this note should not expect to get all the answers at this point. Only if/when this approach is accepted, or at least thought to be potentially useful, will the next stage of analysis be undertaken.

10.2 BACKGROUND

Archives have, and will have, different capabilities.

Some repositories will not be able, for one reason or another, to be compliant with the OAIS-IF requirements.

What is the minimum that a repository must be able to do i.e., what cannot be done within the framework but that the framework needs in order to “fill in the gaps”?

Given that we are basing this OAIS-IF approach on the OAIS Reference Model then one would think that the Framework needs to know:

How do I (the Framework) get the data object?

How do I get its Representation Information?

How do I get its Provenance Information?

How do I get its Access Rights Information?

How do I get its Fixity Information?

How do I get its Context Information?

How do I get its Reference Information?

10.3 ANALYSIS

These questions are of two types:

How to get a Data Object

How to get an Information Object, which involves getting:

a. The Data Object of that Information Object (see point 1) plus
b. The Representation Information of that Information Object (go back to the start of point 2)

Going back to the questions one might reasonably imagine then the range of answers could include:

- Use this URL
- Use this identifier in a call to Glacier

- Use this TCP/IP socket and use this protocol
- Use ABC (a generic name used below)
- Etc

Of course the answer may be "I have no way for you to get these things" e.g. if the archive does not keep Provenance. In programming terms this would be equivalent to returning a NULL. We would then need to decide whether or not to accept NULL. Or perhaps we could accept NULL in some cases e.g. OK not to have CONTEXT, but must have PROVENANCE. This might be regarded as a PROFILE for interoperability for an archive, which one finds in some communication standards.

Bear in mind that Provenance Information, for example, could be as simple as an ASCII English text file which says "I do not have any Provenance". Until we have a way to evaluate the Provenance (in this example) then this could be regarded as the default answer if we receive NULL, and would not be any different from the archive creating this trivial text message and returning it to the OAIS-IF instead of NULL.

10.4 REPRESENTATION INFORMATION INTERFACES

The Negotiation step which is described in the document defined in section 6.4 is to ensure that enough Representation Information is provided. This therefore means that the system should be able to recognize whether or not the is enough of the various types of Representation Information, and how to get more of the Representation Information Network if required. The Representation Information Interface document defines the required interfaces.

The initial idea is to collect summaries of approaches to encoding Representation Information (RI) and then try to extract commonalities, following this a general interface will be drafted.

10.4.1 OAIS DEFINITIONS

***Representation Information**: The information that maps a Data Object into more meaningful concepts so that the Data Object may be understood in ways exemplified by Preservation Objectives.*

NOTE: An example of Representation Information for a bit sequence which is a FITS file might consist of the FITS standard which defines the format plus a dictionary which defines the meaning in the file of keywords which are not part of the standard. This would then allow the information in the FITS file to be used by a computer program to display the image which may be contained in the FITS file, together with the associated coordinate system so that a human can identify objects of interest, for example stars or galaxies. Alternatively, the computer program may identify such objects automatically.

***Representation Information Network**: The set of Representation Information that fully describes the meaning of a Data Object. Representation Information in digital forms needs additional Representation Information so its digital forms can be understood over the Long Term.*

Semantic Information: *The Representation Information that further describes the meaning of the Data Object beyond that provided by the Structure Information.*

Structure Information*: The Representation Information that imparts meaning about how the Data Object is organized.*

> *NOTE: For example, Structure Information maps bit streams to common computer types such as characters, numbers, and pixels and aggregations of those types such as character strings and arrays.*

Other Representation Information: *Representation Information which cannot easily be classified as Semantic or Structural.*

> *NOTE: For example, software, algorithms, encryption, written instructions and many other things may be needed to understand the Content Data Object in ways exemplified by the Preservation Objectives, all of which therefore would be, by definition, Representation Information, yet would not obviously be either Structure or Semantics. Information defining how the Structure and the Semantic Information relate to each other, or software needed to process a database file would also be regarded as Other Representation Information.*

10.4.2 POSSIBLE REPINFO INTERFACE REQUIREMENTS

An Information Package (IP) will logically contain an Information Object (IO) which consists of the DO and RI. This would allow a user to identify the RI of the DO.
Presumably there are many possible implementations of the IP, and once ingested the components of the IP will be stored in some way in the user's systems.
The implementation details should be hidden by the Interoperability Interface (and associated API(s))
One possibility is for the RI to have as part of it some kind of "manifest" that allows one to cope with Use Cases like 1.2.1, 1.2.2, 1.3.1, 1.3.2, 1.4.1 and 1.4.2.

Also http://int-platform.digitalpreserve.info/wp-content/uploads/sites/5/2014/12/javadoc/model/info/digitalpreserve/interfaces/RepresentationInformation.html

10.5 PROVENANCE INFORMATION INTERFACES

The initial idea is to collect summaries of approached to encoding Provenance and then try to extract commonalities, following this a general interface will be drafted.

OAIS has the following definition:

Provenance Information*: The information that documents the history of the Content Data Object. This information tells the origin or source of the Content Data Object, any changes that may have taken place since it was originated, and who has had custody of it since it was originated. The Archive is responsible for creating and preserving Provenance Information*

from the point of Ingest; however, earlier Provenance Information should be provided by the Producer. Provenance Information adds to the evidence to support Authenticity.
Relevant approaches:

- Open Provenance Model https://openprovenance.org/opm/
- PREMIS Data Model

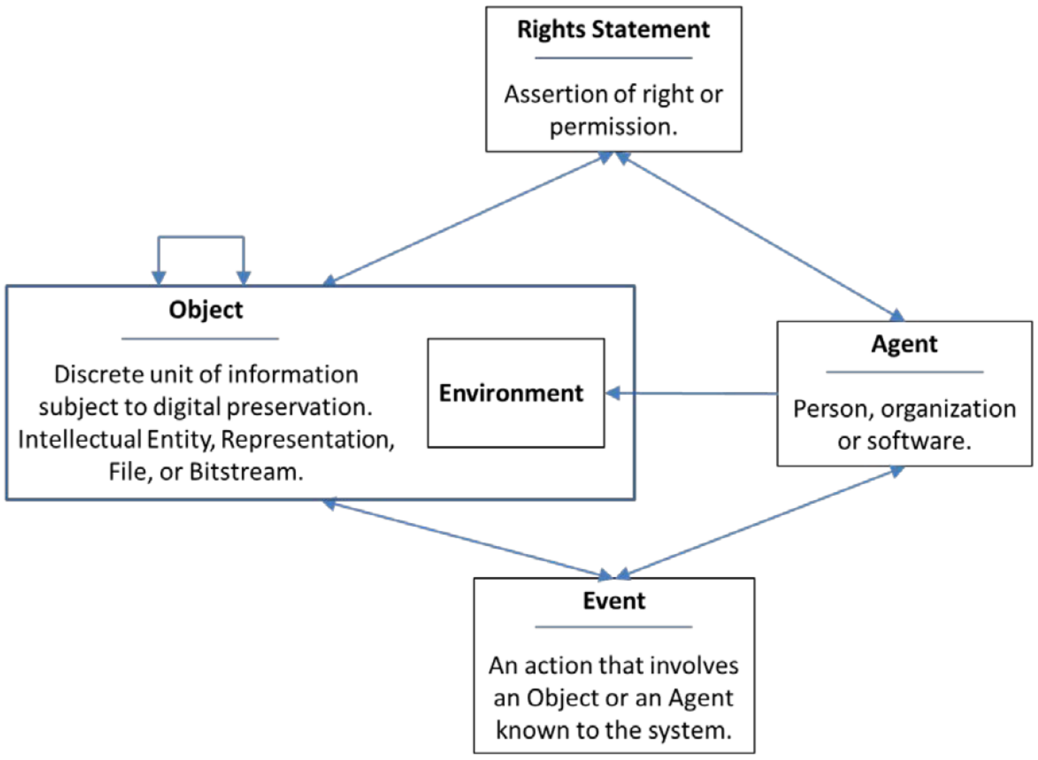
Rights Statement
Assertion of right or permission.
Object
Discrete unit of information subject to digital preservation. Intellectual Entity, Representation, File, or Bitstream.
Environment
Agent
Person, organization or software.
Event
An action that involves an Object or an Agent known to the system.

Figure 10-1 PREMIS Model

Object, Environment, Rights, Agent, Event

- PROV (W3C) https://www.w3.org/TR/prov-overview/

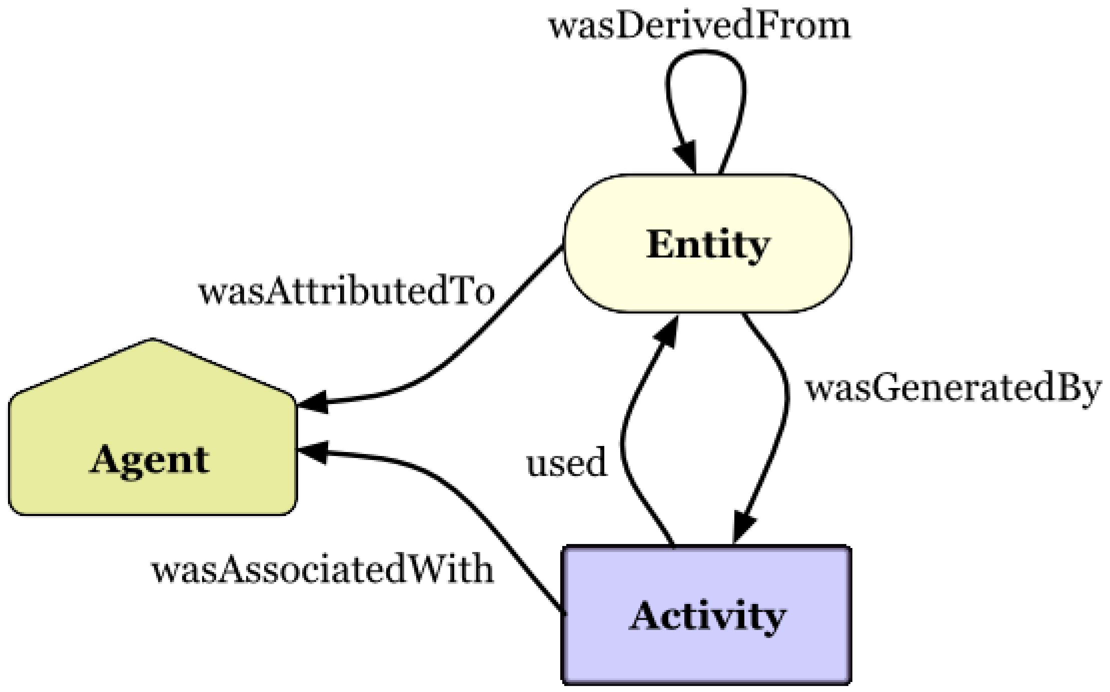

Figure 10-2 PROV Model

Entity, Agent, Activity

- IPAPI - https://www.cl.cam.ac.uk/~acr31/pubs/carata-ipapi.pdf
- Taverna Provenance API - http://www.taverna.org.uk/api/net/sf/taverna/t2/provenance/api/ProvenanceAccess.html

10.5.1 POSSIBLE PROVENANCE INTERFACE REQUIREMENTS

Each of the types of Information identified above will be made up of DO and RI. Representation Information Concept apply.

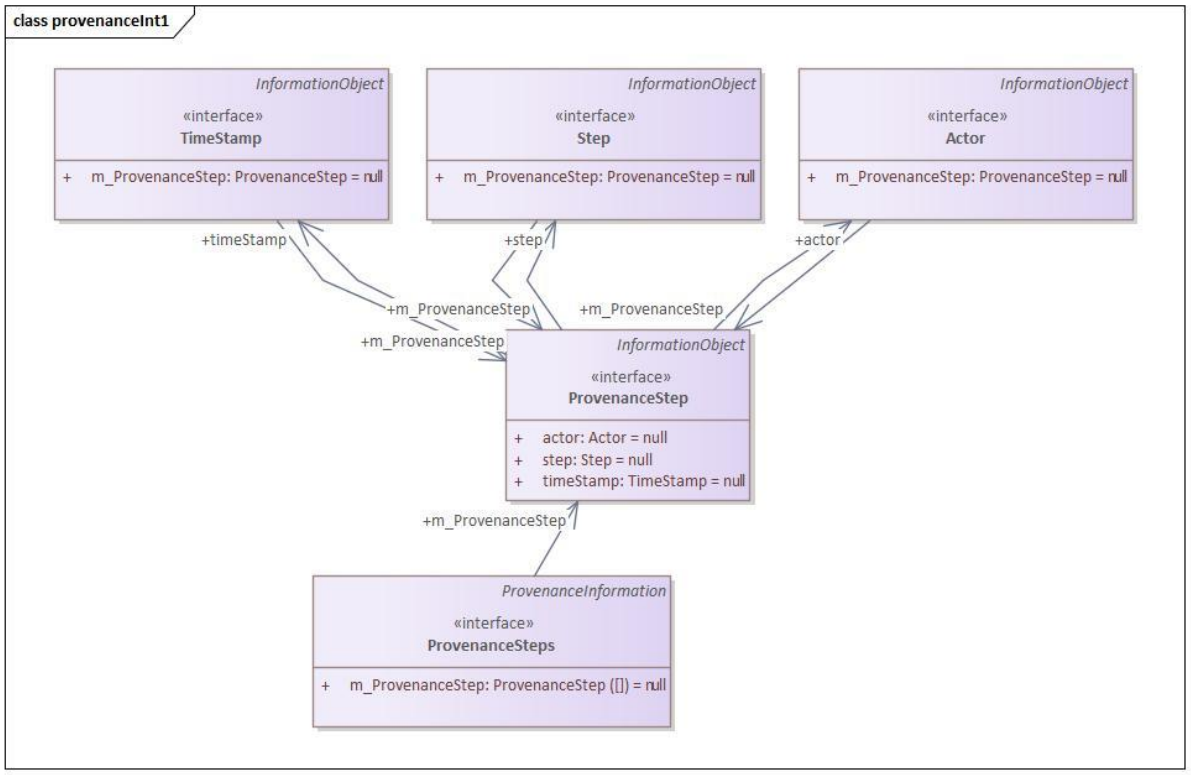

Figure 10-3 possible Provenance interfaces

10.6 FIXITY INFORMATION INTERFACES

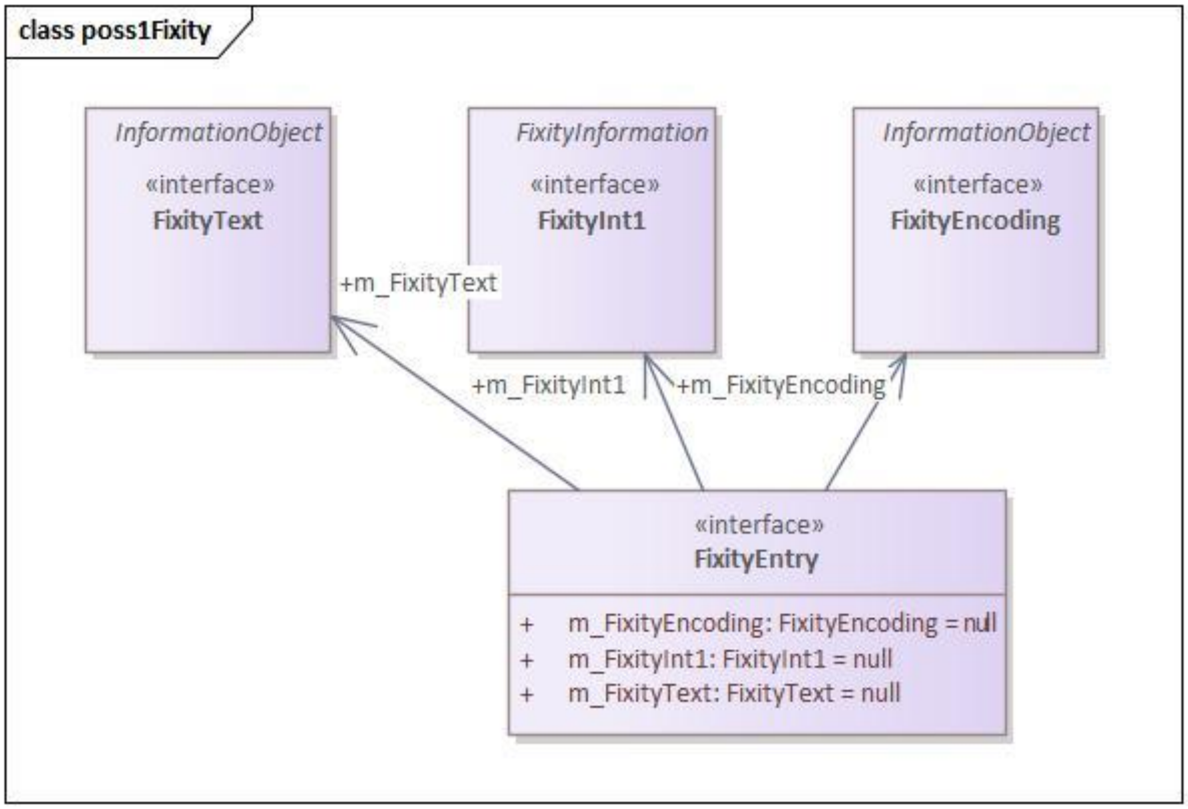

Figure 10-4 possible Fixity interfaces

10.7 ACCESS RIGHTS INFORMATION INTERFACES

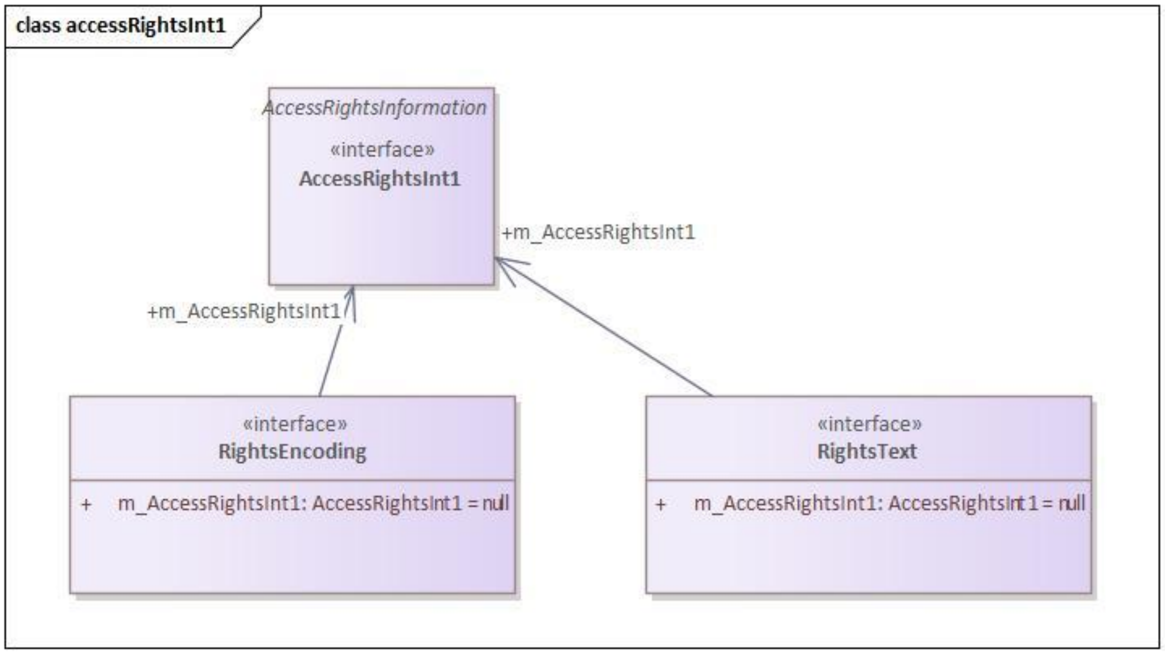

Figure 10-5 possible Access Rights interfaces

10.8 HOLLYWOOD PRINCIPLE

One way to simplify the OAIS-IF design would be to use the Hollywood Principle i.e. “don’t call us, we'll call you”.

In programming design pattern terms this could be an Inversion of Control Pattern or perhaps an Observer Pattern.

As an example, to answer the question “How do I get the Provenance Information for the Data Object which you refer to as (let's say) DOID?” the steps would be:

1. I (the archive) refer to the Provenance Information Object as IOD1
2. Get the Information Object which is Provenance Information as follows:
    a. Use the method ABC with parameters (X, Y, Z) to get the Data Object of the Provenance Information (e.g. a PREMIS file), which I refer to as DOID2, with identifier RIID to get its Representation Information
    b. Use method GHI with parameters (T,U,V) to get the Data Object of the Representation Information (e.g. a description of how to extract elements of Provenance such at event time), which I refer to as DOID3.
        i. Repeat to get as much Representation Information as required [2]
3. If the description allows one to implement a well defined Provenance Interface [1] then the Provenance elements can be accessed programmatically. Otherwise it may require human intervention.

11 REFERENCES

[1] A concept paper on Provenance – see https://docs.google.com/document/d/1YCyhBZKRP7IWhArdI3MnwahjZY5K93UsDZq-cWApYeI/edit?usp=sharing

[2] A concept paper on Representation Information - https://docs.google.com/document/d/1TcYIDKc9WMdyK1POSuitQjdWc5VtlL-YG8Qkt8c8V18/edit?usp=sharing

[3] A concept paper on OAIS-IF ideas – see https://docs.google.com/document/d/14V0wN6nEnG2MaSMmNClRzuDrTxcB0zjv0XNw2pdYDI8/edit#