The Consultative Committee for Space Data Systems

Draft Recommendation for
Space Data System Standards

OAIS-IF CORE SPECIFICATIONS

PROPOSED DRAFT RECOMMENDED STANDARD

CCSDS 000.0-W-0

WHITE BOOK

NOVEMBER 2022

AUTHORITY

Issue:	White Book, Issue 0
Date:	November 2022
Location:	Not Applicable

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

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and represents the consensus technical agreement of the participating CCSDS Member Agencies. The procedure for review and authorization of CCSDS documents is detailed in *Organization and Processes for the Consultative Committee for Space Data Systems* (CCSDS A02.1-Y-4), and the record of Agency participation in the authorization of this document can be obtained from the CCSDS Secretariat at the address below.

This document is published and maintained by:

CCSDS Secretariat
Space Communications and Navigation Office, 7L70
Space Operations Mission Directorate
NASA Headquarters
Washington, DC 20546-0001, USA

FOREWORD

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

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CCSDS shall not be held responsible for identifying any or all such patent rights.

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This Recommended Standard is therefore subject to CCSDS document management and change control procedures, which are defined in *Organization and Processes for the Consultative Committee for Space Data Systems* (CCSDS A02.1-Y-4). Current versions of CCSDS documents are maintained at the CCSDS Web site:

http://www.ccsds.org/

Questions relating to the contents or status of this document should be addressed to the CCSDS Secretariat at the address indicated on page i.

At time of publication, the active Member and Observer Agencies of the CCSDS were:

Member Agencies

- Agenzia Spaziale Italiana (ASI)/Italy.
- Canadian Space Agency (CSA)/Canada.
- Centre National d'Etudes Spatiales (CNES)/France.
- China National Space Administration (CNSA)/People's Republic of China.
- Deutsches Zentrum für Luft- und Raumfahrt (DLR)/Germany.
- European Space Agency (ESA)/Europe.
- Federal Space Agency (FSA)/Russian Federation.
- Instituto Nacional de Pesquisas Espaciais (INPE)/Brazil.
- Japan Aerospace Exploration Agency (JAXA)/Japan.
- National Aeronautics and Space Administration (NASA)/USA.
- UK Space Agency/United Kingdom.

Observer Agencies

- Austrian Space Agency (ASA)/Austria.
- Belgian Federal Science Policy Office (BFSPO)/Belgium.
- Central Research Institute of Machine Building (TsNIIMash)/Russian Federation.
- China Satellite Launch and Tracking Control General, Beijing Institute of Tracking and Telecommunications Technology (CLTC/BITTT)/China.
- Chinese Academy of Sciences (CAS)/China.
- Chinese Academy of Space Technology (CAST)/China.
- Commonwealth Scientific and Industrial Research Organization (CSIRO)/Australia.
- Danish National Space Center (DNSC)/Denmark.
- Departamento de Ciência e Tecnologia Aeroespacial (DCTA)/Brazil.
- European Organization for the Exploitation of Meteorological Satellites (EUMETSAT)/Europe.
- European Telecommunications Satellite Organization (EUTELSAT)/Europe.
- Geo-Informatics and Space Technology Development Agency (GISTDA)/Thailand.
- Hellenic National Space Committee (HNSC)/Greece.
- Indian Space Research Organization (ISRO)/India.
- Institute of Space Research (IKI)/Russian Federation.
- KFKI Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
- Korea Aerospace Research Institute (KARI)/Korea.
- Ministry of Communications (MOC)/Israel.
- National Institute of Information and Communications Technology (NICT)/Japan.
- National Oceanic and Atmospheric Administration (NOAA)/USA.
- National Space Agency of the Republic of Kazakhstan (NSARK)/Kazakhstan.
- National Space Organization (NSPO)/Chinese Taipei.
- Naval Center for Space Technology (NCST)/USA.
- Scientific and Technological Research Council of Turkey (TUBITAK)/Turkey.
- South African National Space Agency (SANSA)/Republic of South Africa.
- Space and Upper Atmosphere Research Commission (SUPARCO)/Pakistan.
- Swedish Space Corporation (SSC)/Sweden.
- Swiss Space Office (SSO)/Switzerland.
- United States Geological Survey (USGS)/USA.

PREFACE

This document is a draft CCSDS Recommended Standard. Its 'White Book' status indicates that its contents are not stable, and several iterations resulting in substantial technical changes are likely to occur before it is considered to be sufficiently mature to be released for review by the CCSDS Agencies.

Implementers are cautioned **not** to fabricate any final equipment in accordance with this document's technical content.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

DOCUMENT CONTROL

Document	Title and Issue	Date	Status
CCSDS 000.0-W-0	OAIS-IF Core Specification Proposed Draft Recommended Standard, Issue 0	October 2022	Current draft

CONTENTS

Table

No table of contents entries found.

1 INTRODUCTION

1.1 [INTRODUCTORY SUBSECTIONS]

[Insert introductory subsections such as PURPOSE, SCOPE, APPLICABILITY, RATIONALE, etc. See CCSDS A20.0-Y-4, *CCSDS Publications Manual* (Yellow Book, Issue 4, April 2014) for the contents of section 1.]

1.2 REFERENCES

The following publications contain provisions which, through reference in this text, constitute provisions of this document. At the time of publication, the editions indicated were valid. All publications are subject to revision, and users of this Recommended Standard are encouraged to investigate the possibility of applying the most recent editions of the publications indicated below. The CCSDS Secretariat maintains a register of currently valid CCSDS publications.

[Only references required for the implementation of the specification are listed in the References subsection. See CCSDS A20.0-Y-4, *CCSDS Publications Manual* (Yellow Book, Issue 4, April 2014) for additional information on this subsection.]

2 OVERVIEW

[Non-normative overview text appears in section 2. See CCSDS A20.0-Y-4, *CCSDS Publications Manual* (Yellow Book, Issue 4, April 2014) for the contents of section 2.]

3 OAIS INFO MODEL - INTERFACES

3.1 INFORMATION OBJECT

Information Object: A Data Object together with its Representation Information. [C1]

The Information Object class is composed of a Data Object and a Representation Information class. This section is normative.

3.1.1 PUBLIC INTERFACE INFOOBJECTINTERFACE

The Information Object Interface is a well-defined entry point for accessing an Information Object. The interface is an element of the Abstraction Layer component.

- All Known Implementing Classes: AccessRightsInformation, ContentInformation, ContextInformation, FixityInformation, InfoObject, PackagedInformation, ProvenanceInformation, ReferenceInformation, RepresentationInformation

3.1.2 METHOD SUMMARY

Modifier and Type	Method and Description
`Object`	`getDataObject()` The getDataObject method returns the Data Object component of this Information Object.
`Identifier`	`getDataObjectID()` The getDataObjectID method returns the identifier of the Data Object component of this Information Object.
`Identifier`	`getInfoObjectID()` The getInfoObjectID method returns the identifier of this Information Object.
`InfoObject`	`getRepInfo()` The getRepInfo method returns the Representation Information component of this Information Object.
`Object`	`getRepInfoDataObject()` The getRepInfoDataObject method returns a Data Object of the Representation Information component of this Information Object.

Identifier	getRepInfoDataObjectID() The getRepInfoDataObjectID method returns the Identifier of the Data Object of the Representation Information for this Information Object.
void	setDataObject(Identifier identifier, DataObject dataObject) The setDataObject method sets the Data Object component of this Information Object.
void	setInfoObjectID(Identifier identifier) The setInfoObjectID method sets the identifier of this Information Object.
void	setRepInfoDataObject(Identifier identifier, RepInfoDataObject repInfoDataObject) The setRepInfoDataObject method sets a Data Object of the Representation Information component of this Information Object.

DATA OBJECT

3.1.2.1 PUBLIC INTERFACE INFOOBJECTINTERFACE

3.1.2.2 Method Summary

Modifier and Type	Method and Description
Identifier	getIdentifier() The getIdentifier method returns the Identifier of this DataObject.
Object	getObject() The getObject method returns the Object for this Data Object
void	setObject(Identifier identifier, Object object) The setObject method sets the object for this Data Object.

3.1.3 SPECIAL INSTANCES

3.1.4 IDENTIFIER

3.1.5 REP INFO

3.1.5.1 Representation_Information_Data_Object

The Representation Information Data Object class implements the Representation Information Data Object Interface and returns Representation Information as a Data Object. In other words the Data Object returned is the representation information component of an Information Object, not Representation_Information as a subclass of Information Object. It is managed by an Archival Storage functional entity and is an element of the OAIS Interoperability Framework component. This section is normative.

3.1.5.1.1 Representation Information Data Object Interface

The Representation Information Data Object Interface is a well-defined entry point for accessing the Data Object of an Information Object's Representation. The interface is an element of the Abstraction Layer component. This section is normative.

- All Known Implementing Classes: RepInfoDataObject

```
public interface RepInfoDataObjectInterface
```

The Representation Information Data Object Interface is a well-defined entry point for getting and putting the Identifier and Object of a Data Object, the Data Object of an Information Object's Representation Information.

1.1.1.1.1.1 **Method Summary**

All Methods	
Modifier and Type	**Method and Description**
`Identifier`	`getIdentifier()` The getIdentifier method returns the Identifier of this RepInfoDataObject.
`Object`	`getObject()`

	The getObject method returns the Object for this RepInfoDataObject
void	setObject(Identifier identifier, Object object) The setObject method sets the Object for this RepInfoDataObject.

1.1.1.1.1.2 **Method Detail**

3.1.5.1.1.1 getIdentifier

```
Identifier getIdentifier()
```

The getIdentifier method returns the Identifier of this RepInfoDataObject.

Returns:
Identifier - the identifier for this repInfoDataObject

3.1.5.1.1.2 getObject

```
getObject()
```

The getObject method returns the Object for this RepInfoDataObject

Returns:
Object - the object for this repInfoDataObject

3.1.5.1.1.3 setObject

```
void setObject(Identifier identifier, Object object)
```

The setObject method sets the Object for this RepInfoDataObject.

Parameters:
`identifier` - the identifier for this repInfoDataObject
`object` - the object for this repInfoDataObject

The Identifier class provides a unique name to an entity to identify it as distinct from other entities.

3.1.5.1.2 Identifier_Interface

The Identifier Interface is a well-defined entry point for accessing the identifier. The interface requires a setAdapter method and is an element of the Abstraction Layer component. This section is normative.

3.2 MESSAGE

The Message class a message is an object, which is sent by a sender, to a receiver.

3.2.1 MESSAGE_INTERFACE

The Message Interface is a well-defined entry point for accessing the components of a Message. The interface is an element of the Adapter Layer component. This section is normative.

- All Superinterfaces:

  InfoObjectInterface

  All Known Implementing Classes: Message

  ```
  public interface MessageInterface

  extends InfoObjectInterface
  ```

  The Message Interface is a well-defined entry point for getting and putting the components of a Message.

  The Message class is defined as an extension of the Info Object class.

  The methods defined are for the local implementation the Message class.

3.2.1.1 Method Summary

All Methods	
Modifier and Type	**Method and Description**
`Identifier`	`getIdentifier()` The getIdentifier method returns the identifier of this message.
`Identifier`	`getReceiverId()`

	The getReceiverId method returns the identifier of the receiver
Object	getReceiverIO() The getReceiverIO method returns the object from the receiver.
Identifier	getSenderId() The getSenderId method returns the identifier of the sender
Object	getSenderIO() The getSenderIO method returns the object from the sender.
void	setIdentifier(Identifier identifier) The setIdentifier method sets the identifier of this message
void	setReceiverId(Identifier receiverId) The setReceiverId method sets the identifier of the receiver
void	setReceiverIO(Object receiverObject) The setReceiverIO method sets the object from the receiver receiver.
void	setSenderId(Identifier senderId) The setSenderId method sets the identifier of the sender receiver.
void	setSenderIO(Object senderObject) The setSenderIO method sets the object from the sender

3.2.1.2 Methods inherited from interface InfoObjectInterface

getDataObject, getDataObjectID, getInfoObjectID, getRepInfo, getRepInfoDataObject, getRepInfoDataObjectID, setDataObject, setInfoObjectID, setRepInfoDataObject

3.2.1.3 Method Detail

3.2.1.3.1 getIdentifier

```
Identifier getIdentifier()
```

The getIdentifier method returns the identifier of this message.

Returns: Identifier the identifier of this message

3.2.1.3.2 getSenderId

```
Identifier getSenderId()
```

The getSenderId method returns the identifier of the sender

Returns: Identifier the identifier of the sender

3.2.1.3.3 getReceiverId

```
Identifier getReceiverId()
```

The getReceiverId method returns the identifier of the receiver

Returns: Identifier the identifier of the receiver

3.2.1.3.4 getSenderIO

```
Object getSenderIO()
```

The getSenderIO method returns the object from the sender.

Returns: Object the Info Object from the sender.

3.2.1.3.5 getReceiverIO

```
Object getReceiverIO()
```

The getReceiverIO method returns the object from the receiver. receiver.

Returns: Object the Info Object from the receiver.

3.2.1.3.6 setIdentifier

```
void setIdentifier(Identifier identifier)
```

The setIdentifier method sets the identifier of this message

Parameters: `identifier` - the identifier of the message

3.2.1.3.7 setSenderId

```
void setSenderId(Identifier senderId)
```

The setSenderId method sets the identifier of the sender receiver.

Parameters: `senderId` - the identifier of the sender

3.2.1.3.8 setReceiverId

```
void setReceiverId(Identifier receiverId)
```

The setReceiverId method sets the identifier of the receiver

Parameters: `receiverId` - the identifier of the receiver

3.2.1.3.9 setSenderIO

```
void setSenderIO(Object senderObject)
```

The setSenderIO method sets the object from the sender

Parameters: `senderObject` - the object being sent from the sender to the receiver

3.2.1.3.10 setReceiverIO

```
void setReceiverIO(Object receiverObject)
```

The setReceiverIO method sets the object from the receiver receiver.

Parameters: `receiverObject` - the object being sent from the receiver to the sender

3.3 REQUEST

The Request class provides is the “Request” messaging interaction protocol where a message is sent to a receiver and a response is expected from the receiver.

3.3.1 REQUEST_INTERFACE

The Request Interface is a well-defined entry point for an interaction protocol that sends a message and receives a response. The interface is an element of the Adapter Layer component. This section is normative.

- All Known Implementing Classes: Request

```
public interface RequestInterface
```

The Request Interface is a well-defined entry point for message enqueue and dequeue.

1.1.1.1.1.3 Method Detail

3.3.1.1 RequestEnqueue

```
void RequestEnqueue(Identifier messageID, Message message)
```

The Request Enqueue method enqueues the message to be sent.

Parameters:
`messageID` - the identifier of the Message to enqueue
`message` - the Message to enqueue

3.3.1.2 RequestDequeue

```
boolean RequestDequeue(Identifier messageID)
```

The RequestDequeue method dequeues the message returned.

Parameters:
`messageID` - the identifier of the Message to dequeue

Returns:
boolean - true if the dequeue was successful

3.4 INFO PACKAGE

Information Package: A logical container composed of optional Content Information and optional associated Preservation Description Information. Associated with this Information Package is Packaging Information used to delimit and identify the Content Information and Package Description information used to facilitate searches for the Content Information. [C1]

The Information Package object class is composed of a Content Information class and a Preservation Description Information (PDI) class. It is a subclass of the Information Object class and inherits its properties. It implements the Information Object Interface, is composed of a Data

Object and Representation Information, is managed by an Archival Storage functional entity, and is an element of the OAIS Interoperability Framework component. This section is normative.

Preservation Description Information (PDI): The information which is necessary for adequate preservation of the Content Information and which can be categorized as Provenance, Reference, Fixity, Context, and Access Rights Information. [C1]

The Preservation Description Information object class is composed of Access Rights Information, Context Information, Fixity Information, Provenance Information, and Reference Information. It provides preservation description for Content Information. It is also a subclass of the Information Object class and inherits its properties. It implements the Information Object Interface, is managed by an Archival Storage functional entity, and is an element of the OAIS Interoperability Framework component. This section is normative.

3.4.1 AIP

3.4.2 REPRESENTATION_INFORMATION

Representation Information: The information that maps a Data Object into more meaningful concepts. 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. Another example is JPEG software which is used to render a JPEG file; rendering the JPEG file as bits is not very meaningful to humans but the software, which embodies an understanding of the JPEG standard, maps the bits into pixels which can then be rendered as an image for human viewing. [C1]

The Representation Information object class implements the Representation Information Interface. The Representation Information class is also a subclass of the Information Object class and inherits its properties. It implements the Information Object Interface, is composed of a Data Object and Representation Information, is managed by an Archival Storage functional entity, and is an element of the OAIS Interoperability Framework component. This section is normative.

3.4.2.1 PDI

3.4.2.2 Provenance_Information

Provenance Information: The information that documents the history of the Content Information. This information tells the origin or source of the Content Information, 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. [C1]

The Provenance Information object class implements the Provenance Information Interface. The Provenance Information class is also a subclass of the Information Object class and inherits its properties. It implements the Information Object Interface, is composed of a Data Object and Representation Information, is managed by an Archival Storage functional entity, and is an element of the OAIS Interoperability Framework component. This section is normative.

3.4.2.3 Reference_Information

Reference Information: The information that is used as an identifier for the Content Information. It also includes identifiers that allow outside systems to refer unambiguously to a particular Content Information. An example of Reference Information is an ISBN. [C1]

The Reference Information object class implements the Reference Information Interface. The Reference Information class is also a subclass of the Information Object class and inherits its properties. It implements the Information Object Interface, is composed of a Data Object and Representation Information, is managed by an Archival Storage functional entity, and is an element of the OAIS Interoperability Framework component. This section is normative.

4 COMMON ADAPTER INTERFACE

The Common Adapter Interfaces contains methods which are common to all adapters.

See WB interface

5 SPECIFIC ADAPTER SPECIFICATION

The Specific Adapter is specific/customized to the User or Archive software. Its function is

(a) to interface to the User/Archive software – these interfaces are not covered in this Standard

(b) to convert the User/Archive software Information Objects into an Information Package serialized as JSON, and pass those to the Generic Adapter, together with information about the target to which the JSON is to be sent

(c) to receive JSON from the Generic Adapter and converts this into Information Objects which can then be used in the User/Archive software.

Some Information Objects are queries, most queries will go to an archive.

Some queries are directed to the **Switchboard** (either Remote specialised archive OR java resource e.g. properties file) to discover how to query e.g. a User to obtain the following information for a specific archive:

- NAME of ARCHIVE
- DESCRIPTION OF ARCHIVE
- URL/PORT
- AUTHENTICATION METHOD FOR COMMS
- SERIALISATION (JSON, XML,,,,) including VERSION
- COMMUNICATION PROTOCOL (REST or MAL or ….)
- QUERY LANGUAGE to use and QUERY PARAMETERS

Some queries are to be directed to a Registry/Repository of Representation Information (**RRORI**) to obtain more RepInfo, if available.

See MB interfaces

6 GENERIC ADAPTER SPECIFICATION

6.1 GENERIC TO SPECIFIC INTERFACES

The GA

(a) receives JSON objects from the SA and sends these to the specified target (User or Archive)

(b) receives the reply, which will be an Information Package, serialized in JSON, and passes that JSON to the SA

6.2 GENERIC TO GENERIC INTERFACES

The Generic Adapter uses the Comms Abstraction Layer to send the message (serialised Information Package over e.g. REST) to the target, which will be a Generic Adapter.

6.3 GENERIC TO SWITCHBOARD INTERFACE

GA queries SWITCHBOARD resource (e.g. properties file, which may also provide a link to a special remote archives which can provide this information) using name of archive to get information specified above), receiving serialized Information Package as follows:

{“name”:” “, “URI”:….}

6.4 GENERIC TO REGISTRY INTERFACE

GA queries REGISTRY resource (e.g. properties file, which may also provide a link to a special remote archives which can provide this information) using description of RepInfo needed to get list of potential RepInfo:

E.g. ask for FITS standard

Require

{[“name”:”FITS v1.0”, “RIOID”:”XYZ”,”DESCRIPTION”:”aksjdl ajs dlak”],

 [……] }

User selects the RepInfo needed e.g. based on constraints of the User systems, and a request is made to retrieve the specific RepInfo.

If no RepInfo is available then {} i.e. NULL, is returned.

Receiving serialized Information Package as follows:

TBD

6.5 GENERALISED SERIALIZATION OF INFORMATION PACKAGE

Serialisation/Deserialisation converts an internal representation of an Information Object to/from a DigitalObject which can be transmitted.

6.6 SPECIFIC SERIALISATION

6.6.1 JSON SERIALISATION OF INFORMATION PACKAGES (GB)

6.6.1.1 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.

A General Information Package would be as follows.

```
{
	"InformationPackage":{
		"PackageType":"General",
		"PackageDescription": " This  is an example IP ",
		"InformationObject":{
			"DataObject":{"IdentifierType":"URI", "Identifier":"http://myprov.example.com/do"},
			"RepInfo":{
				"AndGroup":{
					"SemanticsRI":{ "IdentifierType":"URI",
"Identifier":"http://myprov.example.com/risem"},
					"StructureRI":{ "IdentifierType":"URI",
"Identifier":"http://myprov.example.com/ristr"},
					"OtherRI":{
						"OrGroup": {
							"OtherRI":{"IdentifierType":"URI",
"Identifier":"http://myprov.example.com/rioth-java-sw"},
							"OtherRI":{"IdentifierType":"URI",
"Identifier":"http://myprov.example.com/rioth-csharp-sw"}
						}
					}
				}
			}
		}
	}
}
```

A complete Archival Information Package would be as follows:

```
{
	"InformationPackage":{
		"PackageType":"AIP",
		"PackageDescription": "This  is an example AIP",
		"PDI":{
			"Provenance":{"IdentifierType":"URI","Identifier":"http://myprov.example.com/prov",
"size":"12345678"},
			"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"}
		},
		"InformationObject":{
			"DataObject":{"IdentifierType":"URI",  "Identifier":"http://myprov.example.com/do",
"size":"12345678"},
			"RepInfo":{
```

```
                        "AndGroup":{
                                "SemanticsRI":{                                                    "IdentifierType":"URI",
"Identifier":"http://myprov.example.com/risem", "size":"12345678"},
                                "StructureRI":{                                                    "IdentifierType":"URI",
"Identifier":"http://myprov.example.com/ristr", "size":"12345678"},
                                "OtherRI":{
                                        "OrGroup": {
                                                "OtherRI":{"IdentifierType":"URI",
"Identifier":"http://myprov.example.com/rioth-java-sw"},
                                                "OtherRI":{"IdentifierType":"URI",
"Identifier":"http://myprov.example.com/rioth-csharp-sw"}
                                        }
                                }
                        }
                }
        }
}
```

An Information Package to get a specific InformationObject could be as follows.
Query should have an Identifier so that the response can be associated with it.

```
{
        "InformationPackage":{
                "PackageType":" InfoObjectRequest ",
                " Reference": {" Identifier":"1", " IdentifierType":"Count "},
                "PackageDescription":  "Contains  a  request  for  a  specific  Information  Object  -  identified  by  the
DataObject",
                "InformationObject":{
                        "DataObject":{"IdentifierType":"URI", "Identifier":"http://myprov.example.com/do"},
                        "RepInfo":{"IdentifierType":"URI", "Identifier":""}
                }
        }
}
```

An Information Package for a query could be as follows.
Query should have an Identifier so that the response can be associated with it.

```
{
        "InformationPackage":{
                "PackageType":"Query",
                "Reference": {"Identifier":"1", "IdentifierType":"Count"},
                "PackageDescription": "Contains an Information Object which contains a query",
```

```
              "InformationObject":{
                     "DataObject":{"QueryText":"Get me the latest data on Mars"},
                     "RepInfo":{
                            "AndGroup":{
                                   "SemanticsRI":{ "QueryLanguage":"NaturalEnglishLanguage"},
                                   "StructureRI":{ "Encoding":"ASCII-7"},
                                   "OtherRI":{
                                          "OrGroup": {
                                                 "OtherRI":{"IdentifierType":"URI",                          "Identifier":"
http://natural-language-query.com/rioth-java-sw "},
                                                 "OtherRI":{"IdentifierType":"URI",                          "Identifier":"
http://myprov.example.com/rioth-csharp-sw "}
                                          }
                                   }
                            }
                     }
              }
       }
}
```

An Information Package for an alternative query could be as follows.

```
{
       "InformationPackage":{
              "PackageType":"Query",
              "Reference": {"Identifier":" abcdjjjkakamnaskj", "IdentifierType":"UUID"},
              "PackageDescription": "Contains an Information Object which contains a query",
              "InformationObject":{
                     "DataObject":{"QueryText":" Get ontology I can query "},
                     "RepInfo":{
                            "AndGroup":{
                                   "SemanticsRI":{ "QueryLanguage":"NaturalEnglishLanguage"},
                                   "StructureRI":{"Encoding":"ASCII-7"},
                                   "OtherRI":{
                                          "OrGroup": {
                                                 "OtherRI":{"IdentifierType":"URI",                          "Identifier":"
http://natural-language-query.com/rioth-java-sw "},
                                                 "OtherRI":{"IdentifierType":"URI",                          "Identifier":"
http://myprov.example.com/rioth-csharp-sw "}
                                          }
                                   }
                            }
                     }
              }
```

```
        }
}
```

An Information Package for the response to a query could be as follows.
The response should have the Identifier (“Reference”/”QueryReference”) from the query.

```
{
        "InformationPackage":{
                "PackageType":"QueryResponse",
                "Reference": {"Identifier":"1", "IdentifierType":"Count"},
                "PackageDescription": " Contains an Information Object which contains a query ",
                "InformationObject":{
                        "DataObject":{"IdentifierType":"local", "Identifier":"123456JAGAD"},
                        "RepInfo":{}
                }
        }
}
```

**********TODO**********
Add request for specific component of AIP e.g. Provenance

SWAGGER documentation and testing

Add Packaging version

These examples suggest that we need to define a number of “tags”:

The use of Identifiers is ubiquitous so it seems reasonable to combine the DataObject and RepInfo as exemplified by

{“IdentifierType”:”URI”, “Identifier”:http://natural-language-query.com/rioth-java-sw} i.e.

the DataObject is what follows “Identifier”, while the RepInfo is given by the “IdentifierType”.

PackageType

- AIP – this means that all the AIP components MUST be there
- General – could be anything – need to look at DataObject and RepInfo
- Query – this is a hint that the DataObject is the text of a query, which needs the RepInfo to interpret
- InfoObjectRequest – the object being requested
- QueryResponse – expect an array of identifiers

6.6.1.2 Specialisation of Information Packages (PackageType) e.g. AIP, Query, ..

7 COMMUNICATION MECHANISM

7.1 GENERALISED COMMUNICATION OF INFORMATION PACKAGES

Communication mechanisms transfer DigitalObjects from one system to another.

7.1.1 SPECIFIC COMMS

Specific communications methods are described next.

7.1.1.1 REST (GB)

Reference to general REST

7.1.1.1.1 Look at swagger.io

Error returns:

- Host not found/not available
- JSON cannot be parsed
- Object not found
- Query not understood
- Info Package too large
- …

7.1.1.2 In V2 of BB add MAL

ANNEX A

IMPLEMENTATION CONFORMANCE STATEMENT (ICS) PROFORMA

(NORMATIVE)

A1 INTRODUCTION

A1.1 OVERVIEW

This annex provides the Implementation Conformance Statement (ICS) Requirements List (RL) for an implementation of [Specification]. The ICS for an implementation is generated by completing the RL in accordance with the instructions below. An implementation claiming conformance must satisfy the mandatory requirements referenced in the RL.

A1.2 ABBREVIATIONS AND CONVENTIONS

The RL consists of information in tabular form. The status of features is indicated using the abbreviations and conventions described below.

Item Column

The item column contains sequential numbers for items in the table.

Feature Column

The feature column contains a brief descriptive name for a feature. It implicitly means “Is this feature supported by the implementation?”

Status Column

The status column uses the following notations:

- M mandatory;
- O optional;
- C conditional;
- X prohibited;
- I out of scope;
- N/A not applicable.

Support Column Symbols

The support column is to be used by the implementer to state whether a feature is supported by entering Y, N, or N/A, indicating:

Y Yes, supported by the implementation.

N No, not supported by the implementation.

N/A Not applicable.

The support column should also be used, when appropriate, to enter values supported for a given capability.

A1.3 INSTRUCTIONS FOR COMPLETING THE RL

An implementer shows the extent of compliance to the Recommended Standard by completing the RL; that is, the state of compliance with all mandatory requirements and the options supported are shown. The resulting completed RL is called an ICS. The implementer shall complete the RL by entering appropriate responses in the support or values supported column, using the notation described in A1.2. If a conditional requirement is inapplicable, N/A should be used. If a mandatory requirement is not satisfied, exception information must be supplied by entering a reference X*i*, where *i* is a unique identifier, to an accompanying rationale for the noncompliance.

A2 ICS PROFORMA FOR [SPECIFICATION]

A2.1 GENERAL INFORMATION

ENSURE HEADINGS OF REQIREMENTS CAN BE USED HERE

A2.1.1 Identification of ICS

Date of Statement (DD/MM/YYYY)	
ICS serial number	
System Conformance statement cross-reference	

A2.1.2 Identification of Implementation Under Test

Implementation Name	
Implementation Version	
Special Configuration	
Other Information	

A2.1.3 Identification of Supplier

Supplier	
Contact Point for Queries	
Implementation Name(s) and Versions	
Other information necessary for full identification, e.g., name(s) and version(s) for machines and/or operating systems; System Name(s)	

A2.1.4 Identification of Specification

[CCSDS Document Number]	
Have any exceptions been required? NOTE – A YES answer means that the implementation does not conform to the Recommended Standard. Non-supported mandatory capabilities are to be identified in the ICS, with an explanation of why the implementation is non-conforming.	Yes [ ] No [ ]

A2.2 REQUIREMENTS LIST

[See CCSDS A20.1-Y-1, *CCSDS Implementation Conformance Statements* (Yellow Book, Issue 1, April 2014).]

ANNEX B

SECURITY, SANA, AND PATENT CONSIDERATIONS

(INFORMATIVE)

B1 SECURITY CONSIDERATIONS

B1.1 SECURITY CONCERNS WITH RESPECT TO THE CCSDS DOCUMENT

B1.1.1 Data Privacy

Relies on end points adhering to required privacy

B1.1.2 Data Integrity

DOES REST/TCPIP ensure integrity of message??

B1.1.3 Authentication of Communicating Entities

Rely on REST Authentication

B1.1.4 Control of Access to Resources

Rely on REST Authentication and endpoint authorisation

B1.1.5 Availability of Resources

N/A

B1.1.6 Auditing of Resource Usage

N/A

B1.2 POTENTIAL THREATS AND ATTACK SCENARIOS

- DDOS – relies on endpoint precautions – may result in HTTP error
- SPOOFING – see Authentication
-

B1.3 CONSEQUENCES OF NOT APPLYING SECURITY TO THE TECHNOLOGY

B2 SANA CONSIDERATIONS

Register JSON Schema and UML models (if any)

[See CCSDS 313.0-Y-1, *Space Assigned Numbers Authority (SANA)—Role, Responsibilities, Policies, and Procedures* (Yellow Book, Issue 1, July 2011).]

B3 PATENT CONSIDERATIONS

[See CCSDS A20.0-Y-4, *CCSDS Publications Manual* (Yellow Book, Issue 4, April 2014).]

None