INTERNATIONAL STANDARD

ISO/IEC 21000-8

Second edition
2008-03-01

AMENDMENT 3
2015-04-01

Information technology — Multimedia framework (MPEG-21) —

Part 8: Reference software

AMENDMENT 3: Contract Expression Language (CEL) and Media Contract Ontology (MCO) Reference Software

Technologies de l'information — Cadre multimédia (MPEG-21) —

Partie 8: Logiciel de référence

AMENDEMENT 3: Langage d'expression des contrats (CEL) et ontologie pour contrats de médias (MCO): logiciels de référence

Reference number
ISO/IEC 21000-8:2008/Amd.3:2015(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

This Amendment describes extra reference software for ISO/IEC 21000-20:2013 and ISO/IEC 21000-21:2013.

Information technology — Multimedia framework (MPEG-21) — Part 8: Reference Software, AMENDMENT 3: Contract Expression Language (CEL) and Media Contract Ontology (MCO) Reference Software

In 3.2 Abbreviated terms, insert the following acronyms:

CEL Contract Expression Language

MCO Media Contract Ontology

At the end of 4.2 Overview of ISO/IEC 21000 reference software, add:

— ISO/IEC 21000-20:2013, Information technology — Multimedia framework (MPEG-21) — Part 20: Contract Expression Language (CEL): The corresponding reference software modules provide an environment to manage CEL based documents. A reference to the reference software for ISO/IEC 21000-20:2013 is described in 5.23

— ISO/IEC 21000-21:2013, Information technology — Multimedia framework (MPEG-21) — Part 21: Media Contract Ontology (MCO): The corresponding reference software modules provide an environment to create and edit MCO documents. A reference to the reference software for ISO/IEC 21000-21:2013 is described in 5.24

In Clause 5 Reference software for the ISO/IEC 21000 parts, insert 5.23 and 5.24:

5.23 ISO/IEC 21000-20:2013

5.23.1 Introduction

This Clause describes the reference software for the ISO/IEC 21000-20 Contract Expression Language (CEL).

The CEL Reference Software consists of a number of modules implementing operations and functionalities on Contract documents according to ISO/IEC 21000-20.

5.23.2 CEL Reference Software Java implementation

The Java CEL Reference Software provided implements the following modules:

- *Contract Identification:* This module uniquely identifies an MPEG-21 CEL contract.
  Contract Validation: This module syntactically validates an MPEG-21 CEL contract.
- *Contract Check-with:* This module verifies if a request usage matches with the content expressed in a CEL contract.

A test class for all the modules is provided as well.

5.23.3 Description of CEL Services

The description of the implemented CEL Services is detailed below:

Name or link: Contract Identification
Description
This module allows a user to obtain a uniquely identifier for a contract. The code implemented generates a new unique identifier ID and returns it.
Input
— A CEL contract.
Output
— The unique identifier for the contract.

Name or link: Contract Validation
Description
This module syntactically validates an MPEG-21 CEL contract. The code implemented validates the given contract against the CEL schema.
Input
— A CEL contract.
Output
— It returns if the contract is valid or not and the reasons why.

Name or link: Contract Check With
Description
This service allows users to verify if a request action against a contract matches with its content. The result of the operation includes one of these values: • OK if the action can be performed • USER_NOT_FOUND if the service invoker is different from one of the contract parties • DEONTIC_EXPRESSION_NOT_FOUND if a deontic expression is missing or incomplete • CONDITION_NOT_SATISFIED if the deontic expression is present but exists at least one condition that is not satisfied • UNKNOWN_ERROR if any unknown error occurs The result of the operation also contains a message detailing more information about the result of the operation.
Input
— A CEL contract.
Output

— The result of the contract based authorisation, detailing the reasons why if the authorization has been negative.

5.24 ISO/IEC 21000-21:2013

5.24.1 Introduction

This Clause describes the reference software for the ISO/IEC 21000-21 Media Contract Ontology (MCO).

The MCO Reference Software consists of a number of web applications or services (mco-refsw services) implementing operations and functionalities on Contract documents according to ISO/IEC 21000-21.

5.24.2 mco-refsw services

The mco-refsw set of services can be deployed on an Apache http server, for providing an MCO working environment to a number of users.

Each user can create and operate on MCO documents stored on a dedicated server repository area.

The services are grouped into contexts:

- "MCO CRUD" context - for includes the functionalities for
  - creating, presenting, and editing MCO contract documents
  - exporting MCO documents to the client side
  - deleting MCO documents from the server repository
  - delivering MCO document to a remote server for further delivery to a user
  - loading MCO document to the user working area on server repository
- "Check with and search" context – which includes the functionalities for
  - Check with - for identifying which contracts, in the user repository area, contain Permissions compatible with a user defined target exploitation. A target exploitation is defined by a Permission, for a given user, to act a given action on a given object, under a given set of conditions.
  - Search – for identifying which contracts, in the user repository area, contain the contract elements given in input.
- Help, Info, and Docs context – it is a utility context for making available to the user documentation information on the services.
- Setting context – it is a utility context for allowing the user to set his/her preferences on the behavior of the services.

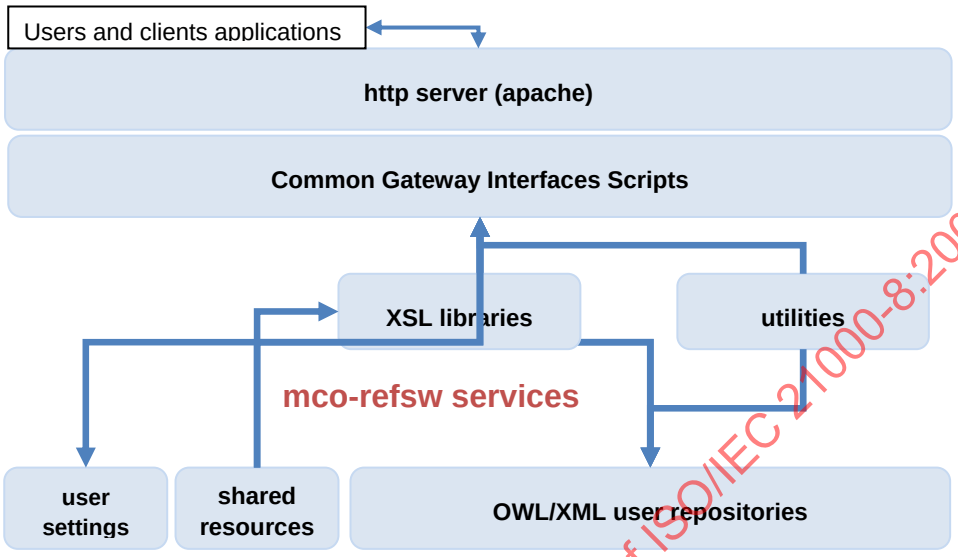
5.24.2.1 Architecture

The technical implementation of "mco-refsw services" is depicted in the Figure below. Services are provided to the users by a normal HTTP server, so that a simple browser can be used, although support of Scalable Vector Graphics (SVG) is required for the Contract Presentation part. The scripts deployed under the common gateway interface parse the user requests and run the basic operations consistently. The actual operations on

the MCO/OWL files, serialized according to OWL/XML specification, are executed by using the provided XSL libraries.

Each User can define his/her settings and owns a part of the document repository.

The “mco-refsw services” can also be used by Client Applications.



5.24.2.2 System requirements

The “mco-refsw” services can be run on any Linux system, although they have been tested specifically on Ubuntu.To install “mco-refsw” on a system with a Linux distribution different from Ubuntu, it is necessary to modify the dependency check part of the configuration and might be useful create aliases to some commands.

The configuration script requires the *tcsh* interpreter.

Before actually installing the software it is required to run the configuration script, which will set a number of properties, the default values of which can be overwritten, by indicating the desired option from the command line.

To run the configuration, execute the following commands from a terminal:

```
$cd /<yourpath>/mco-refsw

$./configure [OPTIONS]
```

For knowing the configuration options and their default values, just run:

```
$./configure -h
```

The configuration script will first run a dependency check. In case some package will result missing, the script will notify the problem and exit.

On success, the configuration will create the scripts for completing the installation and for the users administration.

The installation command must be given with administrative privileges.

```
 $cd /<yourpath>/mco-refsw

$sudo ./install
```

Within the installation, a user named "*default*" is created. You will be required to set a password for this user and confirm it.

The user "*default*" is used also as a template for the creation of other users.

To uninstall it is sufficient to run, as root:

```
$cd /<yourpath>/mco-refsw1

$sudo ./uninstall
```

The uninstall script just makes inactive the services to the http server but it does not remove the users files and the mco-refsw working directories.

5.24.2.2.1 Dependencies for Ubuntu Server 12.04

Dependencies in the case of Ubuntu Server 12.04, out of the box, are:

- tcsh, apache2, xsltproc, graphviz, imagemagic, curl, tofrodos, gridside-clients, libxml2-utils
- Java virtual machine (version 7 or higher) is required for running the java conversion components possibly required by “load” and “export” services. One component, which makes use of OWL-API library (available from http://owlapi.sourceforge.net/), implements OWL serialization conversion from/to OWL/XML to/from RDF/XML, while the other component uses the MCO-to-CEL and CEL-to-MCO conversion modules described in A.15.

5.24.2.3 Description of MCO CRUD

This description provides all the services for “Create, Read, Update, and Delete (CRUD)” MCO documents in the user repository area.

For initiating a new contract document no input is required. A new MCO document, with an automatically generated filename, will be created with an empty contract individual.

By clicking on the ellipse representing the contract individual in graphical representation, the user will get forms for adding other contract elements, such as:

- Contract metadata, such as the contract date;
- Contract Parties, either Organizations or Persons;
- Deontic Expressions (Permissions, Obligations, Prohibitions)
- The textual version of the contract (loading a TXT file)

The refresh button allows the user to re-call the “viewedit” service and get the graphical representations updated with all the added contract elements. The user can get an appropriate set of forms, for each contract element, represented in the diagram with an ellipse, for adding further elements or metadata, or for removing them.

5.24.2.3.1 Main services

Name or link: `http://<host>/cgi-bin/mco-refsw/<user>/mco.crud/create`
Description
Creates a new OWL document, including an Individual of the mco-core:Contract Class.

The default behaviour is that, after successful creation, the OWL document is immediately opened for editing, by re-direction to “viewedit”

The optional input “identifier” is used to name the OWL document and as a suffix to the OntologyIRI prefix, defined in the user settings, for setting the OntologyIRI of the OWL document.

If not given the module will create a default identifier based on the timestamp.

Method

— GET

Input

— identifier - if given it will be the basename of the OWL file, which is otherwise created automatically.

— goedit – Boolean, optional, default=”true”. If true, on Success, the service will directly re-direct to the utility service “viewedit”.

Output

— If (goedit = false)

Text with Resulting code and message for operation: e.g. “`200\nSuccess\n`”

— If (goedit = true) on Success

“`200\nSuccess\n`” and the TEXT/HTML output of “viewedit”

— On failure

Text with Resulting code and message for operation: e.g. “`400\nBad request\n`”

Error codes and message

`400 Bad request` – Method must be GET. Input syntax error

`403 Forbidden` – The user repository area already contains OWL document with the same name

`500 Server Error` – The service could not complete the task successfully

Name or link: `http://<host>/cgi-bin/mco-refsw/<user>/mco.crud/export`

Description

By default it serves the requested OWL file to User, serialized in OWL/XML. If the optional parameter “format” is given, the returned document can also be:

– OWL file serialized in RDF/XML

– CEL / XML Contract, as resulting from the use of the MCO-to-CEL conversion module

Method
— GET
Input
— instance - name of OWL file — format – one among OWL/XML (default), RDF/XML, CEL/XML
Output
— OWL/XML, or RDF/XML, or CEL/XML
Error codes and message
`400 Bad request` – Method must be GET. Input syntax error `404 Not Found` –OWL document not found `500 Server Error` – The service could not complete the task successfully

Name or link: `http://<host>/cgi-bin/mco-refsw/<user>/mco.crud/delete`
Description
Deletes the given OWL file from the User's repository
Method
— GET
Input
— instance - name of OWL file
Output
— Text with Resulting code and message for operation: e.g. “`200\nSuccess\n`”
Error codes and message
`400 Bad request` – Method must be GET. Input syntax error `404 Not Found` –OWL document not found `500 Server Error` – The service could not complete the task successfully

Name or link: http://<host>/cgi-bin/mco-refsw/<user>/mco.crud/deliver

Description

Delivers the given Contract document, stored in User repository area, to a remote service, in order to be further delivered to another User given in input.

This requires the configuration of the URL of the remote service and the account for accessing it, through the settings context.

Method

— GET

Input

— instance - name of OWL file

— deliveruser – User to whom the remote service has to forward the contract

Output

— Text with Resulting code and message for operation: e.g. “`200\nSuccess\n`”

Error codes and message

`400 Bad request` – Method must be GET. Input syntax error

`404 Not Found` –OWL document not found

`500 Server Error` – The service could not complete the task successfully

Name or link: http://<host>/cgi-bin/mco-refsw/<user>/mco.crud/load

Description

Receives the MCO file in input and stores it in the User repository area.

The default behaviour is that, on success, the OWL document is immediately open for editing, by re-direction to “viewedit”.

If input file is not an MCO document with OWL/XML serialization, then conversion is tried from MCO in RDF/XML serialization and from CEL/XML.

Method

— POST

Input

— instance - name of OWL file

— goedit - Boolean, optional, default=”true”. If true, on Success, the service will directly re-direct to the utility service “viewedit”

Output

— If (goedit = false)

Text with Resulting code and message for operation: e.g. “200\nSuccess\n”

— If (goedit = true) on Success

“200\nSuccess\n” and the TEXT/HTML output of “viewedit”

— On failure

Text with Resulting code and message for operation: e.g. “400\nBad request\n”

Error codes and messages

400 Bad request – Method must be POST. Instance might be missing

413 Request Entity Too Large – Size of uploaded file exceeds allowed size (see source for increasing the allowed size)

403 Forbidden – The user repository area already contains OWL document with the same name

500 Server Error – The service could not complete the task successfully

5.24.2.3.2 Other interfaces

Name or link: http://<host>/cgi-bin/mco-refsw/<user>/mco.crud/adddataproperty

Description

Adds a DataProperty, with value, to an Individual of a given OWL instance

Method

— GET

Input

— handle - IRI of Individual to which the DataProperty has to be assigned

— instance - name of OWL file

— key - IRI of the DataProperty

— value - value to be assigned

Output

— Text with Resulting code and message for operation: e.g. “200 Success”

Error codes and message

`400 Bad request` – Method must be GET. Input syntax error `404 Not Found` –OWL document not found `500 Server Error` – The service could not complete the task successfully

Name or link: `http://<host>/cgi-bin/mco-refsw/<user>/mco.crud/adddeontic`
Description
Adds the ObjectProperty mco-core:issuedIn to the Individual of Contract handle from an Invidual of DeonticExpression ind, which is created if not existing or not given
Method
— GET
Input
— class - IRI of the SubClassOf mco-core:DeonticExpression to be added — handle - IRI of Individual of mco-core:Contract to which deontic has to added — ind - IRI of the Deonticexpression — instance - name of OWL file
Output
— Text with Resulting code and message for operation: e.g. “200 Success”
Error codes and message
`400 Bad request` – Method must be GET. Input syntax error `404 Not Found` – OWL document not found `500 Server Error` – The service could not complete the task successfully

Name or link: `http://<host>/cgi-bin/mco-refsw/<user>/mco.crud/addfacts`
Description
Adds a number of Media Contract Facts, together with their DataProperties, to either a DeonticExpression or to a FactComposition Individual. All Parameters are optional. Parameters with value = "null" will be ignored
Method
— GET
Input

— handle - IRI of Individual to which the Facts will be added

— instance - name of OWL file

Classes of Facts

— accesspolicy - IRI of a SubClassOf mco-ipre:AccessPolicy

— deliverymodality - IRI of a SubClassOf mco-ipre:DeliveryModality

— device - IRI of a SubClassOf mco-ipre:Device

— means - IRI of a SubClassOf mco-ipre:Means

— relatedaction - IRI of a SubClassOf mco-core:ActionRelatedFact

— serviceaccesspolicy - IRI of a SubClassOf mco-ipre:ServiceAccessPolicy

— usertimeaccess - IRI of a SubClassOf mco-ipre:UserTimeAccess

Flags for setting Facts to Negative

— negativeaccesspolicy

— negativedeliverymodality

— negativedevice

— negativelanguages

— negativepartofipentitycontext

— negativemeans

— negativerelatedaction

— negativeserviceaccesspolicy

— negativespatialcontext

— negativetemporalcontext

— negativeusertimeaccess

DataProperties implying related Facts

— afterdate - value of mco-ipre:afterDate for a mco-ipre:TemporalContext - format: YYYYMMDD

— beforedate - value of mco-ipre:beforeDate for a mco-ipre:TemporalContext - format: YYYYMMDD

— maxlength - value of mco-ipre:hasMaxLength for a mco-ipre:Length - ISO 8601 Duration

— numruns - value of mco-ipre:hasNumberOfRuns for a mco-ipre:Run - format: positiveInteger

— runduration - value of mco-ipre:hasValidity for a mco-ipre:Run – ISO 8601 Duration

— numreps - value of mco-ipre:hasNumberOfRepetitions for a mco-ipre:Run- format: positiveInteger

— territories - value of mco-ipre:hasCountry for a mco-ipre:SpatialContext - format: list of ISO 3166 country codes (each as '#<CODE>;')

Other inputs implying related Facts:

— partofipentitycontext - IRI of IPEntity to be used as range for a mco-ipre:IPEntityContext fact through ObjectProperty mco-ipre:partOf

Output

— Text with Resulting code and message for operation: e.g. “200 Success”

Error codes and message

`400 Bad request –` Method must be GET. Input syntax error

`404 Not Found –`OWL document not found

`500 Server Error –` The service could not complete the task successfully

Name or link: `http://<host>/cgi-bin/mco-refsw/<user>/mco.crud/addmd`

Description

Add Dublin Core metadata to an Individual of the given OWL instance, as AnnotationProperty

Method

— GET

Input

— handle - IRI of individual to which the metadata will be added

— instance - name of OWL file

— mdkey - metadata field to add

— mdvalue - value of the metadata field

Output

— Text with Resulting code and message for operation: e.g. “200 Success”

Error codes and message

`400 Bad request –` Method must be GET. Input syntax error

`404 Not Found –`OWL document not found

`500 Server Error` – The service could not complete the task successfully

Name or link: http://<host>/cgi-bin/mco-refsw/<user>/mco.crud/addobj
Description
Adds an ObjectProperty obj from the Individual handle to the Individual ind. If ind does not exist or is not given, it will be created as member of Class class.
Method
— GET
Input
— class - IRI of the Class to which the Individual acting as range of ObjectProperty belongs — handle - IRI of Individual acting as domain of ObjectProperty — ind - IRI of Individual acting as range of ObjectProperty — instance - name of OWL file — obj - IRI of the Object Property to be added
Output
— Text with Resulting code and message for operation: e.g. “200 Success”
Error codes and message
`400 Bad request` – Method must be GET. Input syntax error `404 Not Found` –OWL document not found `500 Server Error` – The service could not complete the task successfully

Name or link: http://<host>/cgi-bin/mco-refsw/<user>/mco.crud/addtextualclause
Description
Adds ind as Individual of mco-core:TextualClause, with its text as DataProperty, as the range of the ObjectProperty mco-core:implements having as domain an Individual of DeonticExpression handle. If ind does not exist or is not given, it is created.
Method
— POST
Input
— clausetext - value of the mco-core:Text DataProperty

— handle - IRI of Individual of mco-core:DeonticExpression acting as domain

— ind - IRI of Individual of mco-core:TextualClause acting as range

— instance - name of OWL file

Output

— Text with Resulting code and message for operation: e.g. “200 Success”

Error codes and message

`400 Bad request` – Method must be GET. Input syntax error

`404 Not Found` –OWL document not found

`500 Server Error` – The service could not complete the task successfully

Name or link: `http://<host>/cgi-bin/mco-refsw/<user>/mco.crud/definepermission`

Description

It completely defines a Permission for an Action, including all required Facts and their DataProperties. If the given handle is of Class mvco:Permission, the permitted Action will be created, but the IPEntity over which it can be acted will still need to be identified. If the given handle is of Class mvco:IPEntity (or its SubClasses), both the Permission and the Action will be created and thus the deontic expression will be completely defined..

Method

— GET

Input

— handle - IRI of Individual from which the request was activated. If it is of Class mvco:IPentity (or its subclasses), a new mvco:Permission will be created, otherwise the handle is expected to be an undefined Permission

— instance - name of OWL file

— permittedaction - IRI of a SubClassOf mvco:Action, which is going to be permitted by the defined Permission

DataProperties of the Permission

— isexclusive - boolean value of mco-ipre:isExlclusive (default: false)

— hassublicense - boolean value of mco-ipre:hasSublicenseRight (default: true)

Classes of Facts

— accesspolicy - IRI of a SubClassOf mco-ipre:AccessPolicy

— deliverymodality - IRI of a SubClassOf mco-ipre:DeliveryModality

— device - IRI of a SubClassOf mco-ipre:Device

— means - IRI of a SubClassOf mco-ipre:Means

— relatedaction - IRI of a SubClassOf mco-core:ActionRelatedFact

— serviceaccesspolicy - IRI of a SubClassOf mco-ipre:ServiceAccessPolicy

— usertimeaccess - IRI of a SubClassOf mco-ipre:UserTimeAccess

Flags for setting Facts to Negative

— negativeaccesspolicy

— negativedeliverymodality

— negativedevice

— negativelanguages

— negativemeans

— negativepartofipentitycontext

— negativerelatedaction

— negativeserviceaccesspolicy

— negativespatialcontext

— negativetemporalcontext

— negativeusertimeaccess

DataProperties implying related Facts

— afterdate - value of mco-ipre:afterDate for a mco-ipre:TemporalContext - format: YYYYMMDD

— beforedate - value of mco-ipre:beforeDate for a mco-ipre:TemporalContext - format: YYYYMMDD

— languages - value of mco-ipre:hasLanguage for a mco-ipre:Language - format: list of ISO 639-1 language codes (each as '#<code>;')

— maxlength - value of mco-ipre:hasMaxLength for a mco-ipre:Length - ISO 8601 Duration

— numruns - value of mco-ipre:hasNumberOfRuns for a mco-ipre:Run - format: positiveInteger

— runduration - value of mco-ipre:hasValidity for a mco-ipre:Run – ISO 8601 Duration

— numreps - value of mco-ipre:hasNumberOfRepetitions for a mco-ipre:Run- format: positiveInteger

— territories - value of mco-ipre:hasCountry for a mco-ipre:SpatialContext - format: list of ISO

3166 country codes (each as '#<CODE>;')

Other inputs implying related Facts:

– partofipentitycontext - IRI of IPEntity to be used as range for a mco-ipre:IPEntityContext fact through ObjectProperty mco-ipre:partOf

Output

— Text with Resulting code and message for operation: e.g. “200 Success”

Error codes and message

`400 Bad request` – Method must be GET. Input syntax error

`404 Not Found` –OWL document not found

`500 Server Error` – The service could not complete the task successfully

Name or link: `http://<host>/cgi-bin/mco-refsw/<user>/mco.crud/loadtextversion`

Description

Uploads the textual version of a Contract to be attached as DataProperty to the given Individual of mco-core:Contract

Method

— POST

Input

— contracttextversion - upload plain/text of contract

— handle - IRI of Individual of Contract

— instance - name of OWL file

Output

— Text with Resulting code and message for operation: e.g. “200 Success”

Error codes and message

`400 Bad request` – Method must be POST. Input syntax error

`404 Not Found` –OWL document not found

`500 Server Error` – The service could not complete the task successfully

Name or link: http://<host>/cgi-bin/mco-refsw/<user>/mco.crud/mergein
Description
Tries to merge two Indivuals (of same Class) into a single one. The remaining Individual will inherits all the Properties of the removed one. This service can be used when it is recognised that two distinct Individuals of the same Class should actually be the same.
Method
— GET
Input
— handle - IRI of Individual to merge — instance - name of OWL file — mergetarget - IRI of the Individual target of the merge
Output
— Text with Resulting code and message for operation: e.g. “200 Success”
Error codes and message
`400 Bad request –` Method must be GET, Input syntax error `404 Not Found –`OWL document not found `500 Server Error –` The service could not complete the task successfully

Name or link: http://<host>/cgi-bin/mco-refsw/<user>/mco.crud/remove
Description
Removes the given Individual, all its Properties (Object and Data), and any remaining "orphans" from the given OWL file, where orphans are those Individuals without any ObjectProperty.
Method
— GET
Input
— handle - IRI of Individual to be removed — instance - name of OWL file
Output
— Text with Resulting code and message for operation: e.g. “200 Success”
Error codes and message

`400 Bad request` – Method must be GET. Input syntax error `404 Not Found` –OWL document not found `500 Server Error` – The service could not complete the task successfully

Name or link: `http://<host>/cgi-bin/mco-refsw/<user>/mco.crud/removedataproperty`
Description
Removes the DataProperty, identified by its IRI, its value, and its owner, from the given OWL file.
Method
— GET
Input
— handle - IRI of Individual owning the DataProperty to be removed — instance - name of OWL file — key - IRI of the DataProperty to be removed — value - value of the DataProperty to be removed
Output
— Text with Resulting code and message for operation: e.g. “200 Success”
Error codes and message
`400 Bad request` – Method must be GET. Input syntax error `404 Not Found` –OWL document not found `500 Server Error` – The service could not complete the task successfully

Name or link: `http://<host>/cgi-bin/mco-refsw/<user>/mco.crud/removemd`
Description
Removes the AnnotationProperty, identified by its IRI, its value, and its owner, from the given OWL file.
Method
— GET
Input

- handle - IRI of Individual owning the AnnotationProperty to be removed
- instance - name of OWL file
- mdkey - IRI of the AnnotationProperty to be removed
- mdvalue - value of the AnnotationProperty to be removed

Output

- Text with Resulting code and message for operation: e.g. “200 Success”

Error codes and message

`400 Bad request` – Method must be GET. Input syntax error

`404 Not Found` –OWL document not found

`500 Server Error` – The service could not complete the task successfully

Name or link: `http://<host>/cgi-bin/mco-refsw/<user>/mco.crud/showmd`

Description

Shows the AnnotationProperties of the given Individual as html output. Foreach AnnotationProperties a form for requesting its removal is also served.

Method

- GET

Input

- handle - IRI of Individual the AnnotationProperties of which are showed
- instance - name of OWL file

Output

- Text/HTML

Error codes and message

`400 Bad request` – Method must be GET. Input syntax error

`404 Not Found` –OWL document not found

`500 Server Error` – The service could not complete the task successfully

Name or link: http://<host>/cgi-bin/mco-refsw/<user>/mco.crud/showtextversion
Description
Shows the TextualVersion of the given Individual of mco-core:Contract as html output. A form for requesting its removal is also served.
Method
— GET
Input
— handle - IRI of Individual of mco-core:Contract the TextualVersion of which is showed — instance - name of OWL file
Output
— Text/HTML
Error codes and message
`400 Bad request` – Method must be GET. Input syntax error `404 Not Found` –OWL document not found `500 Server Error` – The service could not complete the task successfully

5.24.2.4 Description of Check-with

Name or link: http://<host>/cgi-bin/mco-refsw/<user>/mco.query/checkwith
Description
Creates a query, in terms of a target exploitation, according to the given parameters and compare it with the indexed Permissions found in the contract documents of the User repository, returning the list of MCO documents having Permissions matching the target exploitation. The Permission does not need to be exactly the same, but must be compatible with target one, i.e. it must either the same or wider (having less conditions, or conditions with wider boundaries, or more general permitted actions).
Method
— GET
Input
— output – optional, default="html", used to set the output format — owner - IRI of who (User or Organization) acts the permitted action(s). Optional, if missingm this will not be considered in the query and any User might match — IPEntity - IRI of the Object of the Permission. Optional, if missing, it will not be

considered in the query and any IPEntity might match.

— relid - Identifier of the Object of the Permission. Optional, considered only if also IPEntity is given, otherwise any Identifier might match.

— permittedaction - IRI of a SubClassOf mvco:Action, which is going to be permitted by the defined Permission. Mandatory

DataProperties of the Permission

— isexclusive - boolean value of mco-ipre:isExlclusive (default: false)

— hassublicense - boolean value of mco-ipre:hasSublicenseRight (default: true)

Classes of Facts

— accesspolicy - IRI of a SubClassOf mco-ipre:AccessPolicy

— deliverymodality - IRI of a SubClassOf mco-ipre:DeliveryModality

— device - IRI of a SubClassOf mco-ipre:Device

— means - IRI of a SubClassOf mco-ipre:Means

— serviceaccesspolicy - IRI of a SubClassOf mco-ipre:ServiceAccessPolicy

— usertimeaccess - IRI of a SubClassOf mco-ipre:UserTimeAccess

Flags for setting Facts to Negative

— negativeaccesspolicy

— negativedeliverymodality

— negativedevice

— negativelanguages

— negativemeans

— negativerelatedaction

— negativeserviceaccesspolicy

— negativespatialcontext

— negativeusertimeaccess

DataProperties implying related Facts

— afterdate - value of mco-ipre:afterDate for a mco-ipre:TemporalContext - format: YYYYMMDD

— beforedate - value of mco-ipre:beforeDate for a mco-ipre:TemporalContext - format: YYYYMMDD

— languages - value of mco-ipre:hasLanguage for a mco-ipre:Language - format: list of ISO

639-1 language codes (each as '#<code>;')

— numruns - value of mco-ipre:hasNumberOfRuns for a mco-ipre:Run - format: positiveInteger

— runduration - value of mco-ipre:hasValidity for a mco-ipre:Run – ISO 8601 Duration

— numreps - value of mco-ipre:hasNumberOfRepetitions for a mco-ipre:Run- format: positiveInteger

— territories - value of mco-ipre:hasCountry for a mco-ipre:SpatialContext - format: list of ISO 3166 country codes (each as '#<CODE>;')

Output

— if (output="html" and On Success)

Text/HTML providing for each matching MCO document

- Name of the MCO document
- Icon of the diagram with its graphical representation
- Button for getting a large version of the diagram in a new window
- Button for calling the "viewedit" service for the MCO document

— if (output !="html") and On Success

Text with the list of MCO document names matching the check-with

Error codes and message

`400 Bad request` – Method must be GET. Input syntax error

`500 Server Error` – The service could not complete the task successfully

Name or link: `http://<host>/cgi-bin/mco-refsw/<user>/mco.query/dosearch`

Description

Search for contract documents in the User repository containing MCO elements matching the given search criteria for MCO Individuals (member of MCO classes). All the inputs are optional and are all considered in AND.

Method

— GET

Input

— output – optional, default="html", used to set the output format

— class - IRI of the Class

- ind - IRI of an Individual
- md – text of annotation
- mdexact - Boolean, default=”false”. If “true” the service will look for exact match of the annotation fields, otherwise it will look for annotation fields simply containing the string given in “md”
- key – IRI of the DataProperty
- value – Value of the DataProperty
- valueexact - Boolean, default=”false”. If “true” the service will look for exact match of the data property values, otherwise it will look for data property values simply containing the string given in “value”
- obj – IRI of the ObjectProperty. If given, the service will look for individuals having this ObjectProperty either as domain or range, depending of “objtype”
- objtype – one between “domain” and “range”. Default is “range”
- negative – Boolean, default=”false. If true the service will look for NegativeObjectProperties instead of ObjectProperties.

Output

- if (output=”html” and On Success)

  Text/HTML providing for each matching MCO document
  - Name of the MCO document
  - Icon of the diagram with its graphical representation
  - Button for getting a large version of the diagram in a new window
  - Button for calling the “viewedit” service for the MCO document
- if (output !=”html”) and On Success

  Text with the list of MCO document names matching the search

Error codes and message

`400 Bad request` – Method must be GET. Input syntax error

`500 Server Error` – The service could not complete the task successfully

Add A.13, A.14 and A.15:

A.13 ISO/IEC 21000-20:2013

Annex A.13 describes the utility software for ISO/IEC 21000-20.

The Java CEL utility software provided implements the following modules:

- *Contract Storage:* This module stores a contract expressed in MPEG-21 CEL format.
- *Contract Delivery:* This module sends a contract to a Service Provider in order to be further delivered to one user.
- *Contract Search:* This module allows users to search contracts by different criteria.

A test for all the utility classes has been provided. The context of the utility modules is detailed below:

Name or link: Contract Storage
Description
This service implements the functionality needed to store a contract. The code saves the contract in the local file system. It is just an example of how the service can be implemented. Any specific implementation must consider the architecture of the contract repository used.
Input
— A CEL contract.
Output
— It returns if the contract has been successfully stored or not, providing in this case the reasons why.

Name or link: Contract Delivery
Description
This module allows a Service Provider to deliver a contract to one user. No deliver protocol is defined to perform the action, so the specific delivery protocol must be defined by the Service Provider in order to perform the delivery. The existing implementation represents a simple http delivery of a contract to a user.
Input
— A CEL contract.
Output
— It returns if the contract has been successfully delivered to the peer or not, providing, if not, the reasons why.

Name or link: Contract Search
Description
This service implements the functionality needed to search contracts according to a set of terms and conditions provided by a user. The search can be performed by matching the following parameters: • Text in the contract • Contract Identifier • Party • IPEntity against which the contract applies