DSPACE-CRIS

https://cineca.github.io/dspace-cris

Technical Documentation

DSPACE-CRIS

Technical Documentation

Draft version 0.9
2014-01-19

Table of Contents

Chapter

1

DSpace-CRIS: introduction

Universities and researcher centers are rethinking their communication strategies, highlighting the quality of their research output and the profiles of their best researchers. Listing publications from an Expert Finder system may represent a solution, but providing an Expert Finder system within an Institutional Repository (IR) is a more innovative approach. This idea was developed in 2009 by the University of Hong Kong Libraries, along with Cineca technicians, and applied to their IR, The HKU Scholars Hub at http://hub.hku.hk/, powered by DSpace.

In 2013, Cineca and HKU went one step further and released DSpace-CRIS, an open source general solution to enrich DSpace with CRIS entities and concepts. "A Current Research Information System, commonly known as CRIS, is any informational tool dedicated to provide access to and disseminate research information." (www.eurocris.org)

At institutional level, a CRIS is a tool for policy making, evaluation of research based on outputs, documenting research activities and output and assistance in project planning and constitutes a formal record of research in progress. For the individual end users, a CRIS is essential to evaluate opportunities for research funding, avoid duplication of research activity, analyze trends, have references to full text or multimedia scholarly publications, locate new contacts and identify new markets for products of research. Typical forms of output are researcher CV, management information, reports to funders, research bibliography and commercial output reports.

DSpace-CRIS consists of a data model describing objects of interest to Research and Development and a set of tools to manage the data. Standard DSpace used to deal with publications and data sets, whereas DSpace-CRIS involves other CRIS entities: Researcher Pages, Projects, Organization Units and Second Level Dynamic Objects (single entities specialized by a profile, such as Journal, Prize, Event, etcetera; because any profile can define its own set of properties and nested objects).

DSpace-CRIS comply with the CERIF[1] standard indeed the key components of the CERIF Data Model are supported natively in DSpace-CRIS: universally unique identifiers (UUID), time stamped relations, semantic characterization.

The flexibility of the DSpace-CRIS data model allows the Institutions to configure the system in several different ways, so that the level of compliance with CERIF may depend on the specific configuration adopted by the Institution. Some de-normalizations are sometimes even recommended, because they are easier to adopt

---

[1] http://www.eurocris.org/Index.php?page=CERIFreleases&t=1

at the project start-up, when data are already available in other systems even if they are not enough structured (i.e. Journal information stored in the publication record or funding information stored in the project record).

DSpace-CRIS supports interoperability through SOAP WebServices for read only access to CRIS information, import from XML and export CRIS entities in CERIF XML 1.6.

DSpace-CRIS: a brief functional description

While DSpace just allows to manage publications, DSpace-CRIS permits to handle other entities such as projects, people, departments, etc., via UI.
Every entity data structure is configurable via UI (User Interface) by adding simple or complex fields. Of course the system also allows to manage new relations among entities.
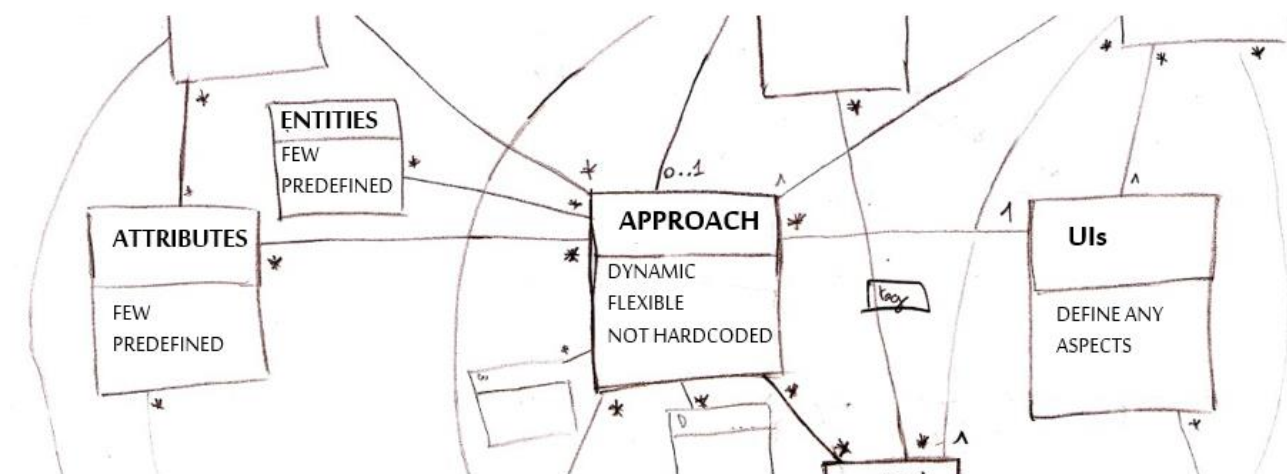
Once the data model has been configured, all entities may have a proper public page, where some or every information may be shown, and they may be searched and browsed.
The DSpace-CRIS entities can be used as authority file for publication's metadata (dspace items), thus producing manageable lists for Authors, Journals, Events, Projects, Funders, etc.
Any relation between a DSpace item and a DSpace-CRIS entity or among DSpace-CRIS entities can be automatically explored in the inverse manner so that it can be produced inside the Researcher Page a list of publications authored by the researcher, or the list of researcher members of the organization, and so on.
Among other advantages, DSpace-CRIS has a unique, unambiguous way to assert that a publication (i.e. a dspace item) is related to a person or to a project without rely on the mere "string value" of the metadata, thus solving the problem of the homonym attribution issues and allowing the building of more advanced functionalities such as:

- aggregation of statistics at different levels: publication lists for researchers, projects for researchers, researchers for OrgUnits, etcetera;
- graphically display and analyze various relationship networks such as co-authorship, collaborative projects, departmental interactions, etcetera.

Moreover, in order to manage information such as the publication citation index, the analysis of usage statistics, the public statistics, and so on, it provides an automatic data retrieving service from external systems, actually from PubMed.

Logical and physical data model

In order to manage the persistence of the "dynamic registries" DSpace-CRIS embrace JDynA[2], an open source JAVA library whose aim is to provide JPA persistence for such dynamic structure.
The DDL of the DSpace-CRIS database can be found here:
https://github.com/Cineca/DSpace/blob/dspace-cris-master/dspace/etc/postgres/addon-crismodule.sql

The core entities of JDynA are shown in the following diagram.
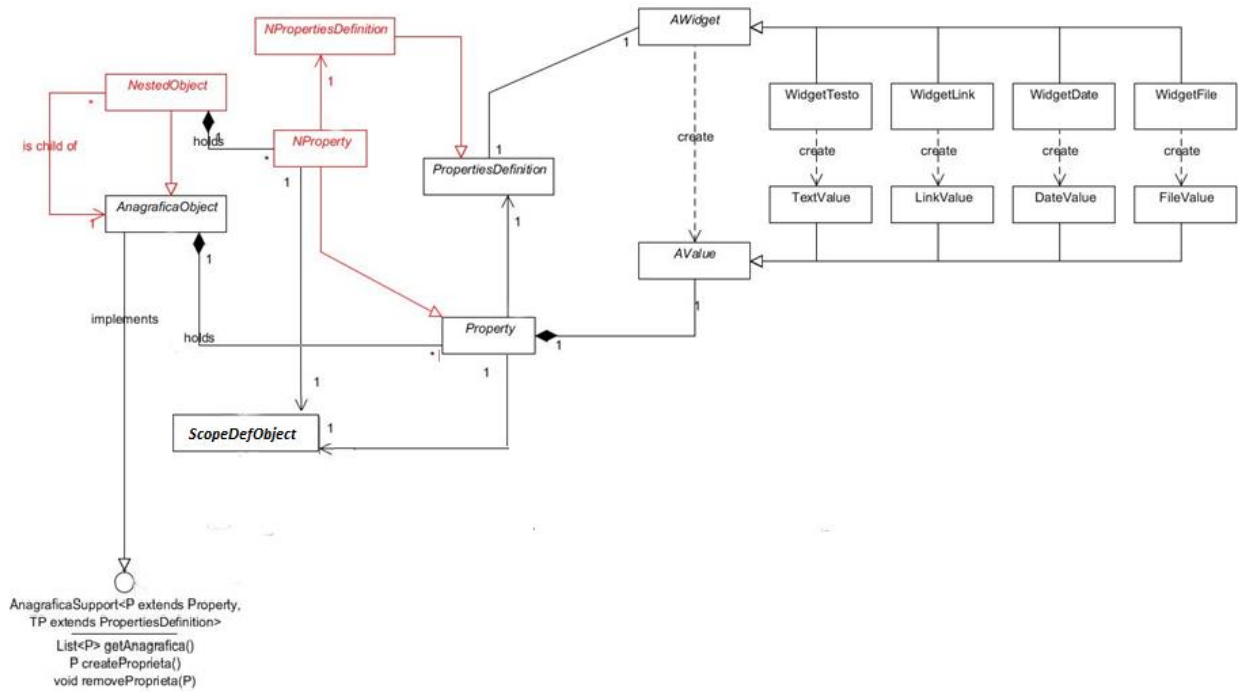
Abstract class AnagraficaObject is inherited by all the objects with dynamic registry. Property class contains both the field definition (PropertiesDefinition) and its value (AValue). AWidget is a framework inner class born to instantiate the correct data types (String, Data, etc.) and to manage the data input modes (autocomplete, dropdown, etc.).
The diagram does not show all the data types that may be handled. It is also possible to support numeric values, links to other entities, Boolean data, classifications, opened or closed subjects list.
JDynA may be extended, in order to create other data types. At the end, the DB structure can be seen as a richer key-value storage.
The following E-R diagram shows the DSpace-CRIS core entities.

---

[2] https://github.com/Cineca/JDynA/wiki
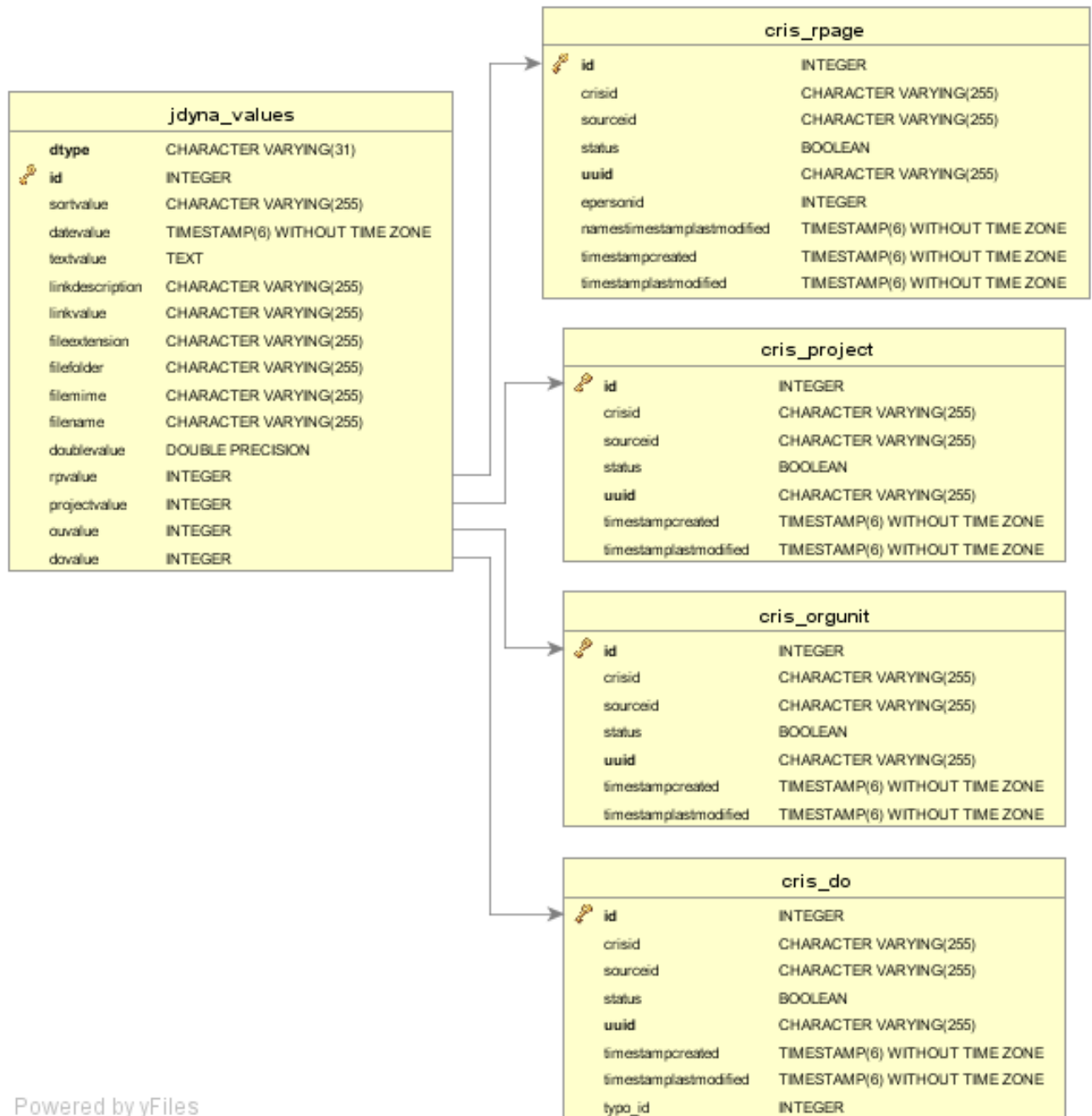
JDYNA_VALUES is included in the core entities diagram because it collects the value of all the information type managed by DSpace-CRIS (in particular the references to system entities: RPVALUE, PROJECTVALUE, OUVALUE and DOVALUE). Thanks to JDYNA_VALUES DSpace-CRIS allows to manage unlimited relations among any object within the module.

CRIS_DO table represents the "Dynamic Object", later renamed "Research Object". It permits to handle all the research entities dedicated to dissemination information process, such as labs, instruments, awards,

etc. In fact JDynA supports the definition of object whose configuration depends on the typology (different set of fields for labs, awards, etc.)

The following E-R diagram shows the tables related to OrgUnit, ResearchPage (the equivalent of CERIF Person) and Project entities.

cris_ou_no
id	INTEGER
enddate	TIMESTAMP(6) WITHOUT TIME ZONE
startdate	TIMESTAMP(6) WITHOUT TIME ZONE
positiondef	INTEGER
status	BOOLEAN
timestampcreated	TIMESTAMP(6) WITHOUT TIME ZONE
timestamplastmodified	TIMESTAMP(6) WITHOUT TIME ZONE
uuid	CHARACTER VARYING(255)
scopded_id	INTEGER
parent_id	INTEGER
typo_id	INTEGER

jdyna_scopedefinition
id	INTEGER
label	CHARACTER VARYING(255)

cris_rpage
id	INTEGER
crisid	CHARACTER VARYING(255)
sourceid	CHARACTER VARYING(255)
status	BOOLEAN
uuid	CHARACTER VARYING(255)
sponsorid	INTEGER
namea timestamplastmodified	TIMESTAMP(6) WITHOUT TIME ZONE
timestampcreated	TIMESTAMP(6) WITHOUT TIME ZONE
timestamplastmodified	TIMESTAMP(6) WITHOUT TIME ZONE

cris_project
id	INTEGER
crisid	CHARACTER VARYING(255)
sourceid	CHARACTER VARYING(255)
status	BOOLEAN
uuid	CHARACTER VARYING(255)
timestampcreated	TIMESTAMP(6) WITHOUT TIME ZONE
timestamplastmodified	TIMESTAMP(6) WITHOUT TIME ZONE

cris_do
id	INTEGER
crisid	CHARACTER VARYING(255)
sourceid	CHARACTER VARYING(255)
status	BOOLEAN
uuid	CHARACTER VARYING(255)
timestampcreated	TIMESTAMP(6) WITHOUT TIME ZONE
timestamplastmodified	TIMESTAMP(6) WITHOUT TIME ZONE
typo_id	INTEGER

cris_do_tp
id	INTEGER
label	CHARACTER VARYING(255)
shortname	CHARACTER VARYING(255)

cris_orgunit
id	INTEGER
crisid	CHARACTER VARYING(255)
sourceid	CHARACTER VARYING(255)
status	BOOLEAN
uuid	CHARACTER VARYING(255)
timestampcreated	TIMESTAMP(6) WITHOUT TIME ZONE
timestamplastmodified	TIMESTAMP(6) WITHOUT TIME ZONE

jdyna_values
dtype	CHARACTER VARYING(31)
id	INTEGER
sortvalue	CHARACTER VARYING(255)
datevalue	TIMESTAMP(6) WITHOUT TIME ZONE
textvalue	TEXT
linkdescription	CHARACTER VARYING(255)
linkvalue	CHARACTER VARYING(255)
fileextension	CHARACTER VARYING(255)
filefolder	CHARACTER VARYING(255)
filemime	CHARACTER VARYING(255)
filename	CHARACTER VARYING(255)
doublevalue	DOUBLE PRECISION
rpvalue	INTEGER
projectvalue	INTEGER
ouvalue	INTEGER
dovalue	INTEGER

cris_ou_no_prop
id	INTEGER
enddate	TIMESTAMP(6) WITHOUT TIME ZONE
startdate	TIMESTAMP(6) WITHOUT TIME ZONE
lockdef	INTEGER
positiondef	INTEGER
visibility	INTEGER
scopded_id	INTEGER
value_id	INTEGER
parent_id	INTEGER
typo_id	INTEGER

cris_ou_no_pdef
id	INTEGER
accesslevel	INTEGER
advancedsearch	BOOLEAN
fieldmin_col	INTEGER
fieldmin_col_unit	CHARACTER VARYING(255)
fieldmin_row_unit	CHARACTER VARYING(255)
fieldmin_row	INTEGER
help	TEXT
label	CHARACTER VARYING(255)
labelminsize	INTEGER
labelminsizeunit	CHARACTER VARYING(255)
mandatory	BOOLEAN
newline	BOOLEAN
oncreation	INTEGER
priority	BOOLEAN
repeatable	BOOLEAN
shortname	CHARACTER VARYING(255)
simplesearch	BOOLEAN
showinlist	BOOLEAN
rendering_id	INTEGER

cris_ou_pdef
id	INTEGER
accesslevel	INTEGER
advancedsearch	BOOLEAN
fieldmin_col	INTEGER
fieldmin_col_unit	CHARACTER VARYING(255)
fieldmin_row_unit	CHARACTER VARYING(255)
fieldmin_row	INTEGER
help	TEXT
label	CHARACTER VARYING(255)
labelminsize	INTEGER
labelminsizeunit	CHARACTER VARYING(255)
mandatory	BOOLEAN
newline	BOOLEAN
oncreation	INTEGER
priority	BOOLEAN
repeatable	BOOLEAN
shortname	CHARACTER VARYING(255)
showinlist	BOOLEAN
simplesearch	BOOLEAN
rendering_id	INTEGER

cris_ou_prop
id	INTEGER
enddate	TIMESTAMP(6) WITHOUT TIME ZONE
startdate	TIMESTAMP(6) WITHOUT TIME ZONE
lockdef	INTEGER
positiondef	INTEGER
visibility	INTEGER
scopdef_id	INTEGER
value_id	INTEGER
parent_id	INTEGER
typo_id	INTEGER

cris_ou_no_tp
id	INTEGER
label	CHARACTER VARYING(255)
shortname	CHARACTER VARYING(255)
accesslevel	INTEGER
help	CHARACTER VARYING(4000)
inline	BOOLEAN
mandatory	BOOLEAN
newline	BOOLEAN
priority	INTEGER
repeatable	BOOLEAN

cris_ou_no_tp2pdef
cris_ou_no_tp_id	INTEGER
cris_ou_no_pdef_id	INTEGER

CRIS_OU_PDEF, CRIS_OU_NO_PDEF and CRIS_OU_NO_TP tables contain the semantic of the organization unit defined by the institution. In JDynA the object semantic is persisted in the data structure.

The following E-R diagram shows the widget JDynA structure used by DSpace-CRIS.

jdyna_widget_text

id	INTEGER
collisioni	BOOLEAN
widgetcol	INTEGER
measurementunitcol	CHARACTER VARYING(255)
measurementunitrow	CHARACTER VARYING(255)
widgetrow	INTEGER
htmltoolbar	CHARACTER VARYING(255)
multilinea	BOOLEAN
regex	CHARACTER VARYING(255)

cris_rp_wfile

id	INTEGER
filedescription	TEXT
labelanchor	CHARACTER VARYING(255)
showpreview	BOOLEAN
widgetsize	INTEGER
useinstatistics	BOOLEAN

cris_pj_wfile

id	INTEGER
filedescription	TEXT
labelanchor	CHARACTER VARYING(255)
showpreview	BOOLEAN
widgetsize	INTEGER
useinstatistics	BOOLEAN

cris_ou_wfile

id	INTEGER
filedescription	TEXT
labelanchor	CHARACTER VARYING(255)
showpreview	BOOLEAN
widgetsize	INTEGER
useinstatistics	BOOLEAN

jdyna_widget_number

id	INTEGER
max	DOUBLE PRECISION
min	DOUBLE PRECISION
precisiondef	INTEGER
widgetsize	INTEGER

jdyna_widget_link

id	INTEGER
labelheaderlabel	CHARACTER VARYING(255)
labelheaderurl	CHARACTER VARYING(255)
widgetsize	INTEGER

jdyna_widget_date

id	INTEGER
maxyear	INTEGER
minyear	INTEGER
time	BOOLEAN

The following E-R diagram shows the configuration tables of OrgUnit entity.

cris_ou_tab

id	INTEGER
ext	CHARACTER VARYING(255)
mandatory	BOOLEAN
mime	CHARACTER VARYING(255)
priority	INTEGER
shortname	CHARACTER VARYING(255)
title	CHARACTER VARYING(255)
visibility	INTEGER

cris_ou_etab

id	INTEGER
ext	CHARACTER VARYING(255)
mandatory	BOOLEAN
mime	CHARACTER VARYING(255)
priority	INTEGER
shortname	CHARACTER VARYING(255)
title	CHARACTER VARYING(255)
visibility	INTEGER
displaytab_id	INTEGER

cris_ou_tab2box

cris_ou_tab_id	INTEGER
cris_ou_box_id	INTEGER

cris_ou_etab2box

cris_ou_etab_id	INTEGER
cris_ou_box_id	INTEGER

cris_ou_box2con

cris_ou_box_id	INTEGER
jdyna_containable_id	INTEGER

cris_ou_box

id	INTEGER
collapsed	BOOLEAN
externaljsp	CHARACTER VARYING(255)
priority	INTEGER
shortname	CHARACTER VARYING(255)
title	CHARACTER VARYING(255)
unrelevant	BOOLEAN
visibility	INTEGER

jdyna_containable

dtype	CHARACTER VARYING(31)
id	INTEGER
externaljsp	CHARACTER VARYING(255)
cris_rp_pdef_fk	INTEGER
cris_pj_pdef_fk	INTEGER
cris_ou_pdef_fk	INTEGER
cris_rp_no_pdef_fk	INTEGER
cris_pj_no_pdef_fk	INTEGER
cris_ou_no_pdef_fk	INTEGER
cris_rp_no_tp	INTEGER
cris_pj_no_tp	INTEGER
cris_ou_no_tp_fk	INTEGER
cris_do_no_tp_fk	INTEGER
cris_do_no_pdef_fk	INTEGER
cris_do_pdef_fk	INTEGER

jdyna_box_message

id	INTEGER
body	CHARACTER VARYING(4000)
showinedit	BOOLEAN
showinpublicview	BOOLEAN
usebodyaskeymessagebundle	BOOLEAN
elementafter_id	INTEGER
parent_id	INTEGER

cris_ou_no_tp

id	INTEGER
label	CHARACTER VARYING(255)
shortname	CHARACTER VARYING(255)
accesslevel	INTEGER
help	CHARACTER VARYING(4000)
inline	BOOLEAN
mandatory	BOOLEAN
newline	BOOLEAN
priority	INTEGER
repeatable	BOOLEAN

cris_ou_no_pdef

id	INTEGER
accesslevel	INTEGER
advancedsearch	BOOLEAN
fieldmin_col	INTEGER
fieldmin_col_unit	CHARACTER VARYING(255)
fieldmin_row_unit	CHARACTER VARYING(255)
fieldmin_row	INTEGER
help	TEXT
label	CHARACTER VARYING(255)
labelminsize	INTEGER
labelminsizeunit	CHARACTER VARYING(255)
mandatory	BOOLEAN
newline	BOOLEAN
oncreation	BOOLEAN
priority	INTEGER
repeatable	BOOLEAN
shortname	CHARACTER VARYING(255)
showinlist	BOOLEAN
simplesearch	BOOLEAN
rendering_id	INTEGER

cris_ou_no_tp2pdef

cris_ou_no_tp_id	INTEGER
cris_ou_no_pdef_id	INTEGER

cris_ou_pdef

id	INTEGER
accesslevel	INTEGER
advancedsearch	BOOLEAN
fieldmin_col	INTEGER
fieldmin_col_unit	CHARACTER VARYING(255)
fieldmin_row_unit	CHARACTER VARYING(255)
fieldmin_row	INTEGER
help	TEXT
label	CHARACTER VARYING(255)
labelminsize	INTEGER
labelminsizeunit	CHARACTER VARYING(255)
mandatory	BOOLEAN
newline	BOOLEAN
oncreation	BOOLEAN
priority	INTEGER
repeatable	BOOLEAN
shortname	CHARACTER VARYING(255)
showinlist	BOOLEAN
simplesearch	BOOLEAN
rendering_id	INTEGER

Powered by yFiles

The configuration process allows defining tab to display the entity data and their groupings (box).

About the CERIF compliance

> ⚠ The CERIF export is not yet released. This documentation describes the approach that we are using for the feature development that will be released in the next version (for both DSpace 3.x and DSpace 4.x).

Since the actual data model is effectively build upon the specific need of the institution through the configuration process, the actual degree of compliance with the CERIF data model can vary. The system provides a general infrastructure that allow to translate different configurations to the CERIF data model, see "Configure the CERIF Mapping". The basic configuration provided out-of-box is a simplified implementation of the key CERIF concepts and entities, it could be extended to support furthers CERIF entities and relationship.

For any CERIF Entity is possible to define one or more corresponding DSpace-CRIS entities (1:N), in this way is for example possible to better characterize a publication from a journal or a grant from the underline project.

Once that the entities mapping CERIF vs DSpace-CRIS has been configured, i.e.:
- People → ResearcherPage
- OrgUnit → OrgUnit
- Publication → DSpace items; Dynamic Object: "Journal"; etc.

The property definition of the DSpace-CRIS Entity can be mapped to the attribute of the CERIF Entity. To manage the multilingual feature of CERIF there are two possible approach:
- The first one, often the most practical, is to define separate property definitions for each languages that the Institution like to support in the system;
- The second one, more accurate, is to use a nested object with two properties one to hold the language and another for the actual value. The language can be stored as simple string holding the ISO code or as a pointer to a dynamic object "ISO Language" where all the available languages are stored. The second one should be preferred where the data is edited via the DSpace-CRIS UI.

To manage the relationship between CERIF entities and specifically the semantic characterization of such relationship there are two possibilities:
- The first one, often the most practical, is to define separate properties definition for each relation meaning so to have, for example, a specific property definition that track the "principal investigator" relation between a project and a people. This mean that for all the relationship meaning that an Insitution want to support a property definition need to be separately configured, i.e. principal investigator, co-investigator, director, member, associate member, etc.
- The second one, more accurate, is to use a nested object with two properties one to hold the relation meaning (semantics) and another for the actual value. The first pointer should be configured to link to the dynamic object "CERIF Semantics Classification" that are providen

out-of-box in the system filtering on a specif CERIF Semantic Schema, i.e. Person role. The second pointer is the link to the target entity of the relation.

For the relationship is usual to use in the same system both approach, keeping the most relevant relationship as separated property definition and tracking the other as nested object. The main reason to make such difference is related to the major flexibility that a separate property definition provides in terms of UI configuration (edit/visualization permission and positioning) and UI customization simplicity.

Finally, the JDynA data model has been extended introducing a ScopeDef object, actually at most a placeholder. The property level holds attributes "startdate", "enddate" (validity dates) and a reference to such object.

Even if these attributes have not been managed by JDynA/DSpace-CRIS yet (they depend on an ad hoc customization), this activity is on the DSpace-CRIS roadmap, and will provide a third way to manage the CERIF mapping in a more natural way. The "scopedef" ref attribute may be used in order to manage multilingual contents or the relation semantics instead of setting specific propertydefinition or use nested objects.

The three different ways to manage the DSpace-CRIS / CERIF mapping allow to choose the better balance between   the complete rappresentation of the research world that CERIF aims to archieve and the data that the Institution really holds or is able to provide maximizing performance and end-users functionalities.

Property Definition, Nested Object and, in future, ScopeDef and validity time stamp attributes will all provide different behaviour in terms of performance[3], UI configuration and indexing.

---

[3] Specifically property are retrieved in eager mode with the object that they belong to; instead, nested objects are loaded in lazy mode when needed. That mean the information that are not alway required for display or indexing work well as nested object than as property from a performance point of view. The extended CERIF attributes (scopedef, start/end date) are expected to be in eager mode with the scopedef objects cached to avoid database query.

Chapter

2

DSpace-CRIS installation

DSpace CRIS is an addon for the DSpace platform. This chapter describes how install the addon on the top of a customized version of DSpace JSPUI, that already includes the "core" changes needed to integrate DSpace CRIS.

The source code of the customized pre-integrated version of DSpace can be found in the Cineca's GitHub Repository.
Current line of development for any DSpace version is maintenad in a branch named dspace-**<version>**-cris, the tagged release dspace-cris-**<version>**

Fresh installation
Current line of development for 3.x.x: https://github.com/Cineca/DSpace/tree/dspace-3_x_x-cris

Tagged releases:

-   dspace-cris-3.2.0-beta : https://github.com/Cineca/DSpace/tree/dspace-cris-3.2.0-beta

DSpace-CRIS Installation Overview (**current line of development for 3.x.x**)

Prerequisite software
See also https://wiki.duraspace.org/display/DSDOC18/Installation#Installation-PrerequisiteSoftware

-- Oracle JDK 6 or 7 or OpenJDK 7
-- git (distributed version control system) - download the git console client for your operative system from http://git-scm.com/
-- Apache Maven 2.2.x or 3.0.x or later (Java build tool) -- suggest the 3.0.4 -- download Maven from http://maven.apache.org/
-- Apache Ant(TM) version 1.7.x or later (Java build tool) -- suggest the 1.8.4 -- download Ant from http://ant.apache.org/
-- Relational database PostgreSQL version 9.x -- suggest the 9.3.1 -- download pgsql from http://www.postgresql.org/
-- Servlet engine Apache Tomcat version 6 or version 7 -- suggest the 6.0.35 or 7.0.42 -- download tomcat from http://tomcat.apache.org/
-- Solr search engine version 4.1.0 -- download from http://archive.apache.org/dist/lucene/solr/4.1.0/solr-4.1.0.zip

0 CREATE THE DSPACE USER (NOT NECESSARY).

This needs to be the same user that Tomcat will run as. e.g. as root run:

```
useradd -m usercris
```

1) GET SOURCES FROM GITHUB REPOSITORY

Cineca organization -- https://github.com/Cineca

```
usercris@servercris /e
$ mkdir crisinstallation

usercris@servercris /e
$ cd crisinstallation/

usercris@servercris /e/crisinstallation
$ git clone https://github.com/Cineca/DSpace.git --branch dspace-3_x_x-cris dspace-parent/
Cloning into 'dspace-parent'...

usercris@servercris /e/crisinstallation
$ cd dspace-parent/

usercris@servercris /e/crisinstallation/dspace-parent (dspace-3_x_x-cris)
$ git submodule sync

usercris@servercris /e/crisinstallation/dspace-parent (dspace-3_x_x-cris)
$ git submodule update --init dspace-cris/
Submodule 'dspace-cris' (https://github.com/Cineca/dspace-cris.git) registered for path 'dspace-cris'
Cloning into 'dspace-cris'...

usercris@servercris /e/crisinstallation/dspace-parent (dspace-3_x_x-cris)
$ cd ..
```

2) DATABASE SETUP

PostgreSQL (if you are an Oracle user please contact us, the addon work on Oracle but the installation procedure is actually uncovered in this documentation):

A PostgreSQL JDBC driver is configured as part of the default DSpace build. You no longer need to copy any PostgreSQL jars to get PostgreSQL installed.
Create a dspace database user. This is entirely separate from the dspace operating-system user created above.

```
createuser -U postgres -d -A -P dspace
```

You will be prompted for the password of the PostgreSQL superuser (postgres). Then you'll be prompted (twice) for a password for the new dspace user.
Create a dspace database, owned by the dspace PostgreSQL user (you are still logged in at 'root'):

```
createdb -U dspace -E UNICODE dspace
```

You will be prompted for the password of the DSpace database user. (This isn't the same as the dspace user's UNIX password.)

3) INITIAL CONFIGURATION:

```
Edit [dspace-source]/build.properties
```

This properties file contains the basic settings necessary to actually build/install DSpace for the first time (see build.properties Configuration for more detail). In particular you'll need to set these properties, examples or defaults are provided in the file:

```
  dspace.install.dir - must be set to the [dspace] (installation) directory  (On Windows be sure to
use forward slashes for the directory path!  For example: "C:/dspace" is a valid path for Windows.)
  dspace.hostname - fully-qualified domain name of web server.
  dspace.baseUrl - complete URL of this server's DSpace home page but without any context eg.
/xmlui, /oai, etc.
  dspace.name - "Proper" name of your server, e.g. "My Digital Library".
```

solr.server - complete URL of the Solr server. DSpace makes use of Solr http://lucene.apache.org/solr/ for indexing purposes.
default.language
db.name - postgres or oracle
db.driver
db.url
db.username - the database password used in the previous step
db.password - the database password used in the previous step.
mail.server - fully-qualified domain name of your outgoing mail server.
mail.from.address - the "From:" address to put on email sent by DSpace.
mail.feedback.recipient - mailbox for feedback mail.
mail.admin - mailbox for DSpace site administrator.
mail.alert.recipient - mailbox for server errors/alerts (not essential but very useful!)
mail.registration.notify- mailbox for emails when new users register (optional)

> ⚠ The "build.properties" file is provided as a convenient method of setting only those configurations necessary to install/upgrade DSpace. Any settings changed in this file, will be automatically copied over to the full "dspace.cfg" file (which is held in [dspace-source]/dspace/config/dspace.cfg).   Refer   to   the   General   Configuration (https://wiki.duraspace.org/display/DSDOC3x/Configuration#Configuration-GeneralConfiguration) section for a fuller explanation.

It is also worth noting that you may choose to copy/rename the "build.properties" under a different name for different environments (e.g. "development.properties", "test.properties", and "production.properties").

You can choose which properties file you want to build DSpace with by passing a "-Denv" (environment) flag to the "mvn package" command (e.g. "mvn package -Denv=test" would build using "test.properties). See General Configuration (https://wiki.duraspace.org/display/DSDOC3x/Configuration#Configuration-GeneralConfiguration) section for more details.

> ⚠ Do not remove or comment out settings in build.properties When you edit the "build.properties" file (or a custom *.properties file), take care not to remove or comment out any settings. Doing so, may cause your final "dspace.cfg" file to be misconfigured with regards to that particular setting.
>
> ⚠ Instead, if you wish to remove/disable a particular setting, just clear out its value. For example, if you don't want to be notified of new user registrations, ensure the "mail.registration.notify" setting has no value e.g.   mail.registration.notify=

4) DSPACE DIRECTORY:

Create the directory for the DSpace installation (i.e. [dspace]). As root (or a user with appropriate permissions), run:

```
mkdir [dspace]
chown dspace [dspace]
```

5) INSTALLATION PACKAGE:

```
usercris@servercris /e/crisinstallation
$ cd dspace-parent/

usercris@servercris /e/crisinstallation/dspace-parent
$ mvn package


[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO] ------------------------------------------------------------------------
[INFO] DSpace Parent Project ............................... SUCCESS [10.408s]
[INFO] DSpace Kernel :: API and Implementation ............... SUCCESS [1:01.623s]
[INFO] DSpace Solr Statistics Logging Client Library ......... SUCCESS [4.127s]
[INFO] Article Metrics :: Modules ........................... SUCCESS [0.007s]
[INFO] Article Metrics :: PubMed ............................ SUCCESS [0.009s]
[INFO] Article Metrics :: PubMed :: Retrieve API ............. SUCCESS [10.846s]
[INFO] DSpace JSP-UI ....................................... SUCCESS [0.007s]
[INFO] DSpace JSP-UI :: API and Implementation ............... SUCCESS [14.494s]
[INFO] Article Metrics :: PubMed :: Lookup API ............... SUCCESS [1.616s]
[INFO] Article Metrics :: PubMed :: Lookup JSP Web ........... SUCCESS [3.366s]
[INFO] DSpace CRIS ......................................... SUCCESS [0.007s]
[INFO] DSpace CRIS :: Web Application Resources .............. SUCCESS [10.324s]
[INFO] DSpace JSP-UI :: Web Application Resources ............ SUCCESS [36.206s]
[INFO] DSpace CRIS :: API .................................. SUCCESS [28.156s]
[INFO] DSpace CRIS :: JSPUI API ............................. SUCCESS [17.557s]
[INFO] DSpace CRIS :: API .................................. SUCCESS [0.011s]
[INFO] CRIS :: WS API ...................................... SUCCESS [9.352s]
[INFO] CRIS webservices-webapp Spring-WS Application ......... SUCCESS [43.629s]
```

```
[INFO] DSpace Addon Modules ................................. SUCCESS [0.008s]
[INFO] DSpace JSP-UI :: Web Application ...................... SUCCESS [2:53.549s]
[INFO] DSpace Assembly and Configuration .................... SUCCESS [1:52.530s]
[INFO] ------------------------------------------------------------------------
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 8 minutes 59 seconds
[INFO] Finished at: Wed Jun 19 17:19:01 CEST 2013
[INFO] Final Memory: 55M/161M
[INFO] ------------------------------------------------------------------------
```

6) BUILD DSPACE AND INITIALIZE DATABASE:

As the dspace UNIX user, initialize the DSpace database and install DSpace to [dspace]_:

```
cd [dspace-source]/dspace/target/dspace-[version]-build
ant fresh_install


-----------------------
BUILD SUCCESSFUL
Total time: 49 seconds
-----------------------
```

7) DEPLOY WEB APPLICATIONS

⚠ Before launch tomcat or deploy war rename webapps to jspui and solr

You copy only (or all) of the DSpace Web application(s) you wish to use from the [dspace]/webapps directory to the appropriate directory in your Tomcat installation. For example:

```
  cp -R [dspace]/webapps/jspui [tomcat]/webapps/ (This will copy only the jspui web application to
Tomcat)
  cp -R [dspace]/webapps/* [tomcat]/webapps* (This will copy all the web applications to Tomcat).
```

> ⚠ Please note that the embedded DSpace Solr is not working with DSpace-CRIS, when copy webapps wipe out this and install a fresh copy of Solr 4.1.0

Deploy Solr 4, download 4.1.0 version and copy war distribution to [tomcat]/webapps, deploy war and configure to recognize your solr home directory. To configure your solr/home you can follow this steps:

1) Manually define the solr home in web.xml (hardcode solr home [dspace]/solr as JNDI property (find it, uncomment and modify the env-entry))

or

2) modify in [tomcat]/conf/context.xml to define a context environment parameter and the others context parameters to override the webapps parameters, for example:

<Environment name="solr/home" value="[dspace.dir]/solr" type="java.lang.String" override="false"/>

Recall, change the [xyz] placeholder with your real path.

> ⚠ If you use **Tomcat 7**, recommended, then you have to change the startup tomcat script to add a system properties that modify the default Tomcat behavior. So startup tomcat with the follow parameter: **-Dorg.apache.el.parser.SKIP_IDENTIFIER_CHECK=true**

8) ADMINISTRATOR ACCOUNT: CREATE AN INITIAL ADMINISTRATOR ACCOUNT:

[dspace]/bin/dspace create-administrator

Administrator account created
----

9) INITIAL STARTUP!

Start up (or restart) Tomcat. Visit the base URL(s) of your server.

Base URL of DSpace Web Applications: JSP User Interface - (e.g.) http://localhost:8080/jspui

You should see the DSpace home page. Congratulations! DSpace-CRIS is live! your journey to the world of DSpace-CRIS is just started!

The DSpace-CRIS sample configuration
By default, the DSpace-CRIS data configuration is empty.

For your convenience we provide a dump of a sample configuration that you can use to explore the system: base-configuration-crismodule.sql:

```
psql -h localhost -d dspace -U dspace -f [dspace]/etc/postgres/base-configuration-crismodule.sql
```

This sql file create for you a base metadata configuration for Researcher profile, Organization Unit and Project entity.
You can build or add custom metadata dynamically using the Admin UI, see "The administrative UI"

Upgrade an existent DSpace installation

If you want to apply DSpace-CRIS on top of an existent DSpace Installation you need to perform a slightly different procedure.

⚠ This procedure assume that you have a working DSpace 3.x version, if you are using an older version please upgrade your dspace installation first using the official DSpace documentation.

⚠ **Backup your content. Backup your content. Backup your content. Backup your content. Backup your content. Backup your content. Backup your content. Backup your content. Backup your content. Backup your content. Backup your content. Backup your content. Backup your content. Backup your content. Backup your content. Backup your content. Backup your content. Backup your content. Backup your content. Backup your content.**

Before you start your upgrade, it is strongly recommended that you create a backup of your DSpace instance. Backups are easy to recover from; a botched install/upgrade is very difficult if not impossible to recover from. The DSpace specific things to backup are: configs, source code modifications, database, and assetstore. On your server that runs DSpace, you might additionally consider checking on your cron/scheduled tasks, servlet container, and database.

You can download the DSpace customizations based on official 3.2 for DSpace-CRIS module:

https://github.com/Cineca/DSpace/archive/dspace-3_x_x-cris.zip

DSpace-CRIS module:

https://github.com/Cineca/dspace-cris/archive/3.x.x.zip

Unzip the DSpace parent project and copy the unzipped content of the DSpace-CRIS module into dspace-cris folder under the root of the parent project. If you are a developer, you are strongly recommended to checkout the Cineca's DSpace fork from GitHub, see "1) get sources from github repository".

For example:

unzip custom DSpace into folder [path-for-dspace-unzipped]\DSpace-dspace-3_x_x-cris (from here dspace-src)

the content of the DSpace-CRIS zip must be copied into folder [dspace-src]\dspace-cris

Now dspace-cris directory looks like with the follow contents:

- api

- jspui-api

- jspui-webapp

- webservices

- other files ...

After that you have to apply all your changes make in the past from your DSpace 3.2 to new DSpace 3.x for DSpace-CRIS

So now you have to build the application from [dspace-src] run the follow command:

```
mvn -U clean package
```

After mvn say build success, you can run "ant update" from [dspace-src]/dspace/target/dspace-[version]-build.dir . The last step is apply all custom folders/files to the [dspace.dir] - refers to the install directory for your existing DSpace installation - and create database table for DSpace-CRIS.

To update your database run:

```
psql -h [db-host] -d [db.name] -U [db.user] -f [dspace-src]/etc/postgres/addon-crismodule.sql
psql -h [db-host] -d [db.name] -U [db.user] -f [dspace-src]/etc/postgres/addon-crispubmed.sql
psql -h [db-host] -d [db.name] -U [db.user] -f [dspace-src]/etc/postgres/addon-subscription.sql
```

Create the following folders in the [dspace.dir]:

```
mkdir rp-files
mkdir rg-files
mkdir ou-files
mkdir do-files
mkdir ws-xsd
```

```
cp [dspace-src]/dspace/etc/webservice/* [dspace.dir]/ws-xsd
```

Now you should be able to deploy your webapps, see "7) Deploy Web Applications" and following steps.

Customize your DSpace-CRIS installation

> ⚠ Discovery is required for DSpace-CRIS to work

Please note that Discovery is the default search provider since DSpace 4.0, whereas in previous releases of DSpace it needs to be manually enabled.
The DSpace-CRIS build of DSpace enables it by default also in 1.8.x, 3.x. **it MUST NOT** be turned off

Browse
DSpace-CRIS extends the DSpace browse system to all the CRIS entities.
This means that the user is able to configure browsing on researchers, organization units, projects and second level dynamic objects such as Journals, Events, Laboratories, and so on that the user has defined.
The configuration syntax follows the same rules of the standard DSpace configuration.
There are two types of browse:

1. Full, single level browse, which just list in a specific order the instances of an Entity class: Researchers, OrgUnits, etc. This type of browse is defined prefixing the shortname of the Class with the word cris

    i.e. crisrp for researchers, crisou for orgunits, crispj for project and finally cris<shortname of the dynamic object> for second level entities
2. Metadata, two levels browse, that provide a first page listing the values of a configured metadata leading to a second page where the instances that have the clicked value for that metadata are listed

Full, single level browse
The syntax to configure full browse (single level) is

```
webui.browse.index.<n> = <index-name>:<display-type>:<sort-name>[:DESC]
```

INDEX-NAME is used to refer to further configurations as the list of columns to show and generate the i18n keys for the navigation (menu links, page header, etc.)

DISPLAY-TYPE can be anything except metadata and metadataAuthority that are reserved word for the two level browse. It can be used to add filters to restrict the set of objects managed by the browse, see "Apply filters to the browse indexes"

SORT-NAME is used to refer to the sorting configuration described below

DESC if used make the descending order the default for that browse

webui.itemlist.sort-option.<n> = <sort-name>:<metadata>:<value-type>

METADATA to give access to the properties of the cris objects using a DSpace like metadata syntax you need to prefix the shortname of the PropertiesDefinition with the shortname of the entity class (same of above, crisrp, crisou, etc.). So if you want to use for sorting the property defined with shortname "myproperty" in researcher entity you need to write crisrp.myproperty
When the property that you are referring to is a pointer you can use the qualifier to access to property of the pointed object, i.e. crisrp.dept.name will be the name of the orgunit assuming that dept is the shortname of a property that keep the relation between the researcher and the orgunits. You can also use virtual metadata, see "ItemEnhancer: virtual metadata".

VALUE-TYPE: can be one of title, text, date or any other alias used to configure a Sort Plugin, see https://wiki.duraspace.org/display/DSDOC3x/Configuration#Configuration-OtherBrowseOptions

To configure which information/columns are displayed for any objects is possible to use the following configuration property

webui.itemlist.<entity >.columns = <field1>(rendering), field2 (rendering), ... fieldN

for example:

webui.itemlist.crisrp.columns = crisrp.fullName(cristitle), crisrp.translatedName, crisrp.email

the configuration defined for a specific entity (crisrp, item, etc.) can be overridden for a specific browse using the following configuration

webui.itemlist.browse.rpname.columns = crisrp.fullName(cristitle), crisrp.email

Metadata, two levels browse
As for the DSpace item it is possible to define browse over a metadata (i.e. authors of an item). In this way the system will produce a two levels browse, the first level will show in a paginated way all the values of the metadata (i.e., all the authors) clicking on a single value will show the list of items that match the selection.

Applying this concept to the CRIS entities, you can for example build a two level browse showing all the departments of the researchers and for any department the list of researchers affiliated

```
webui.browse.index.<n> = <index-name>:metadata:<schema.element.qualifier>:<text|date>
```

for example

```
webui.browse.index.6 = rpdept:metadata:crisrp.dept:text
```

INDEX-NAME is used as reference in the column definition configuration to apply specific configuration for that browse

SCHEMA.ELEMENT.QUALIFIER defines the field upon which the browse is build

TEXT | DATE specify if the values must be interpreted as String or Dates for sorting

Apply filters to the browse indexes

It could be useful to restrict the set of objects for a specific browse applying additional SOLR filter query. To configure a filter for a specific browse you can define the following configuration property

```
browse.solr.bi_<display-type>.filter = <your-solr-filter-query>
```

DISPLAY-TYPE is the value of the second part of the browse configuration. It is metatadata or metadataAuthority for two levels browse or something else for the configuration of a full browse index.

For example

```
browse.solr.bi_item.filter = dateissued:[2000 TO *]
```

It will limit the browse to the items published after the 2000,

```
browse.solr.bi_crisou.filter = crisou.relationwith_authority:"ou00001"
```

It will limit the browse to the OrgUnits that have the field "relationwith" valorized with a pointer to the OrgUnit ou00001.

ItemEnhancer: virtual metadata

The enhancers allow to access information not immediately available in an object (item or cris object) as it was a direct metadata/field of the object. In this way is for example possible to access the item author's affiliation as it was an item metadata.

There are three type of enhancer:

- org.dspace.app.cris.discovery.CrisItemEnhancer, used to add dynamically to the item information from linked CRIS entities as it was metadata of the item

- org.dspace.app.cris.discovery.CrisEnhancer, used to add dynamically to the CRIS entity information from linked CRIS entities as it was metadata of the current CRIS entity

- org.dspace.app.cris.discovery.CrisNestedEnhancer, used to add dynamically to the CRIS entity information from the nested CRIS entities as it was metadata of the current CRIS entity

The enhancers are configured in the spring file **[installDir]/config/spring/cris/cris-metadata-enhancers.xml**

Below are included the snipped of an CrisItemEnhancer in the default configuration

```xml
<bean class="org.dspace.app.cris.discovery.CrisItemEnhancer">
      <property name="alias" value="author" />
      <property name="metadata">
            <list>
                    <value>dc.contributor.author</value>
            </list>
      </property>
      <property name="clazz">
            <value>org.dspace.app.cris.model.ResearcherPage</value>
      </property>
      <property name="qualifiers2path">
            <map>
                    <entry key="dept">
                            <value>dept</value>
                    </entry>
            </map>
      </property>
</bean>
```

The properties have the following meaning

- alias, is the "element" of the virtual field

- qualifiers2path, is a map whose key are the "qualifier" of the virtual field and the value the path, evaluated as EL, in the target object

- metadata, is the list of item metadata to consider that linkage the item with the CRIS object

- clazz, is the canonical name of the linked CRIS Entity class

The previous example will create the virtual metadata crisitem.author.dept that will be evaluated as valorized with the department of the researcher linked to the item through the metadata dc.contributor.author.

The following snippet show a CRISEnhancer and a CRISNestedEnhancer

```
<bean class="org.dspace.app.cris.discovery.CrisEnhancer">
        <property name="clazz">
                <value>org.dspace.app.cris.model.Project</value>
        </property>
        <property name="alias" value="coinvestigators" />
        <property name="qualifiers2path">
                <map>
                        <entry key="dept">
                                <value>dept</value>
                        </entry>
                </map>
        </property>
</bean>
<bean class="org.dspace.app.cris.discovery.CrisNestedEnhancer" scope="prototype">
        <property name="applicationService" ref="applicationService" />
        <property name="nestedClazz" value="org.dspace.app.cris.model.jdyna.ProjectNestedObject" />
        <property name="clazz">
                <value>org.dspace.app.cris.model.Project</value>
        </property>
        <property name="alias" value="grant" />
        <property name="qualifiers2path">
                <map>
                        <entry key="agencies">
                                <value>agencies</value>
                        </entry>
                </map>
        </property>
</bean>
```

The first one allow to use crispj.dept instead of cripj.coinvestigators.dept in this way the field of the target OrgUnit are available for indexing when working on the Project.

The second one allows access to the agencies (the value) of the nested object named grant (the alias) using the metadata crispj.agencies where agencies is the key of the qualifiers2path map.

Search configuration: indexes, facets

The discovery module used by DSpace-CRIS has been extended to be able to manage also CRIS Entities. In the **[installDir]/config/spring/api/discovery-solr.xml** file a new implementation of the SearchService/IndexingService has been defined

```
<bean class="org.dspace.app.cris.discovery.CrisSearchService"
                                        id="org.dspace.discovery.SearchService"/>
```

New special entries can be used in the definition of the DiscoveryConfigurationService in the **[installDir]/config/spring/api/discovery.xml** file to allow specific configuration for entity type:

```
<bean  id="org.dspace.discovery.configuration.DiscoveryConfigurationService"
        class="org.dspace.discovery.configuration.DiscoveryConfigurationService">
        <property name="map">
            <map>
                    <entry key="default" value-ref="defaultConfiguration" />
                    <entry key="site" value-ref="homepageConfiguration" />
                    <entry key="dspacebasic" value-ref="dspaceConfiguration" />
                    <entry key="crisrp" value-ref="crisRPConfiguration" />
                    <entry key="crisproject" value-ref="crisPJConfiguration" />
                    <entry key="crisou" value-ref="crisOUConfiguration" />
                    <!-- <entry key="crisjournal" value-ref="crisDOJournalConfiguration" /> -->
                    <!-- <entry key="123456789/7621" value-ref="defaultConfiguration"/> -->
            </map>
            ...
```

The map containing all the settings, the key is used to refer to the page/scope of the search, the "site", a community/collection handle or an entity type, the value-ref is a reference to a spring bean that actually define the DiscoveryConfiguration format.

DEFAULT is the configuration key used if a not more specific one exist for the current search/indexing scope

DSPACEBASIC is the fallback configuration key for all the standard DSpace Objects: Items, Collections and Community

CRISRP is the configuration key used searching/indexing a ResearcherPage

CRISPJ is the configuration key used searching/indexing a Project

CRISOU is the configuration key used searching/indexing an OrgUnit

CRIS<SHORTNAME> is the configuration key used searching/indexing a Dynamic Object with **<shortname>**, for example journal

The searching scope is defined by the UI implicitly when the search is performed from a "specific page" as a community or collection home page or explicitly when the user choose to restrict the search to a specific subset.

During the indexing phase the scope/configuration key is defined as follow

STANDARD DSPACE OBJECTS such Items, Collections and Communities, the configuration key is the handle of the object if not found the parent object handle will be used as fallback until to the *dspacebasic* or *default* key

CRIS OBJECTS such ResearcherPages, OrgUnits, Projects, Dynamic Objects, the configuration key is dedined the by the entity type with fallback to the *default* key

To configure a DiscoverySearchFilter, DiscoverySearchFilterFacet, DiscoverySortFieldConfiguration the "DSpace like metadata syntax" previously explained, see METADATA in the Browse customization.

The following IndexerPlugins has been added in the **[installDir]/config/spring/api/discovery.xml** file

Additional indexing plugin to implement the CRIS browse system via SOLR

```
<bean id="crisBrowseIndexer" scope="prototype"
      class="org.dspace.app.cris.discovery.CrisBrowseSolrIndexPlugin">
</bean>
```

Additional indexing plugin to implement CRIS relations preferences, see "Relation Preference Management", via SOLR

```
<bean id="crisRelationPreferenceIndexer"
      class="org.dspace.app.cris.discovery.RelationsPreferencesSolrIndexPlugin">
      <property name="applicationService" ref="applicationService" />
</bean>
```

Additional indexing plugin to add bitstream identifier to item SOLR document, this is used by the extended statistics functionalities as bitstream download are associated to the item using SOLR join query, see "Statistics configuration" for further details

```
<bean id="bitstreamIndexer" class="org.dspace.app.cris.discovery.BitstreamSolrIndexer" />
```

Additional indexing plugin to add resource type in human format readable to SOLR document

```
<bean id="resourceTypeIndexer" class="org.dspace.app.cris.discovery.ResourceTypeSolrIndexer" />
```

Additional indexing plugin to add cris authoritylookup for authority framework works with dynamic resource type to SOLR document

```
<bean id="researchObjectAuthorityLookupIndexer"
        class="org.dspace.app.cris.discovery.ResearchObjectAuthorityLookupSolrIndexer />
```

Custom JSPs

To customize the presentation of the data of a CRIS entity other than rearranging the order of the data inside a box and deciding positioning, see "Group Metadata, aka boxes", the system allow you to define your custom JSPs used instead of the out-of-box layout.
You can use custom JSPs for example to add javascripts or automatic linkage for some data (the ORCID, ScopusID, etc.) both in the visualization mode than in edit mode.

> ⚠ In this case, even if the configuration is edited via UI, the layout does not change, because it is defined by the JSP fragment.

CRIS Components

The system allows extension thought the use of component that can extends the presentation, tools and functionalities available for any entities.

The components are registered in the **[installDir]/config/spring/cris/cris-components.xml** spring file. For any CRIS entity type has been defined a service where that manage the components, these services need to be named rpComponentsService, pjComponentsService, ouComponentsService and doComponentsService. The first two letters of the service name refer to the short name of the related CRIS Entity: ResearcherPage (rp), Project (pj), OrgUnit (ou), Dynamic Object (do).

The snippet below show the default configuration for the Researcher Page components Service

```
<!-- Components -->
<bean id="rpComponentsService" class="org.dspace.app.cris.integration.CrisComponentsService">
        <property name="components">
```

```
            <map>
                    <entry key="dspaceitems" value-ref="publicationlist" />
                    <entry key="projects" value-ref="projectlist" />
            </map>
        </property>
</bean>
```

> ⚠ For every components, a box with the same name and a custom JSP file must be defined.
> Placing such box in one or more Tab will enable the components visualization.

Out-of-box the system include configurable components to explore inverse relationship between a CRIS object and DSpace items and a CRIS object and other CRIS objects. This approach other than be time saving make also the data model more consistent avoiding the creation of wrong unilateral relationship. For example: if a researcher belongs to a department it has to be set up a pointer field via UI (see "Field Definition"). However, the OrgUnit entity not need a pointer to track the relationships between the department and all its researchers. The component executes the appropriate SOLR query to retrieve the referencing objects. The snippet below show the configuration of the publicationList component for the ResearcherPage

```
<!-- Publication component for RP -->
<bean id="publicationlist"
        class="org.dspace.app.webui.cris.components.ItemsConfigurerComponent">
        <property name="relationConfiguration" ref="relationRPVSItemConfiguration" />
        <property name="commonFilter">
                <util:constant static-field="org.dspace.app.webui.cris.util.RelationPreferenceUtil.HIDDEN_FILTER"/>
        </property>
        <property name="target" value="org.dspace.app.cris.model.ResearcherPage" />
        <property name="types">
                <map>
                        <entry key="selected" value-ref="selectedComponent" />
                        <entry key="journal" value-ref="journalComponent" />
                        <entry key="conference" value-ref="conferenceComponent" />
                        <entry key="book" value-ref="bookComponent" />
                        <entry key="patent" value-ref="patentComponent" />
                        <entry key="other" value-ref="otherComponent" />
                </map>
        </property>
</bean>
```

org.dspace.app.webui.cris.components.ItemsConfigurerComponent is the implementation of the abstract class org.dspace.app.webui.cris.components.ASolrConfigurerComponent<T extends DSpaceObject, IBC extends ICrisBeanComponent> able to manage relationship between CRIS objects and DSpace Items. The following class provide similar functionalities for the other kinds of relationship

- org.dspace.app.webui.cris.components.CRISDOConfigurerComponent able to explore relationship with Dynamic Object
- org.dspace.app.webui.cris.components.CRISOUConfigurerComponent able to explore relationship with OrgUnit
- org.dspace.app.webui.cris.components.CRISPJConfigurerComponent able to explore relationship with Project
- org.dspace.app.webui.cris.components.CRISRPConfigurerComponent able to explore relationship with ResearcherPage

The following attributes define the behavior of such classes
- relationConfiguration defines a relation between two Entity Class (CRIS Objects or DSpace Item), see the dedicated paragraph for further details
- commonFilter defines an additional SOLR filter query to apply to the relationship
- target must match the class canonical name of the target Entity in the relationship, i.e. if the relationship is Item – ResearcherPage (author) the target class is the ResearcherPage
- types is a map that allow clustering of the relationship under user-defined category (facet query), see the dedicated paragraph for further details

Relationship definition

The relationships are defined in the **[installDir]/config/spring/cris/cris-relationpreference.xml**
A relationConfiguration is shown in the snippet below

```
<bean id="relationRPVSItemConfiguration" class="org.dspace.app.cris.configuration.RelationConfiguration">
        <property name="relationName" value="crisrp.publications" />
        <property name="relationClass" value="org.dspace.content.Item" />
        <property name="query">
                <value>author_authority:{o}</value>
        </property>
</bean>
```

It is defined by
- relationName, an unique name that characterize the relationship
- relationClass, the class canonical name of the Entity Class that holds the main direction of the relationship
- query, the solr query that can be used to retrieve the objects linked to the "current" object. {0} will be replaced with the authority key of the current object

RELATION PREFERENCE MANAGEMENT

The relationConfiguration are used other than by the components by the RelationPreferenceService that can be enabled over one or more relationship so to allow the user to hide some relationship of make a selected

list. The RelationPreferenceService is configured through the definition of a spring bean for the class org.dspace.app.cris.configuration.RelationPreferenceServiceConfiguration the default configuration is shown below

```xml
<bean class="org.dspace.app.cris.configuration.RelationPreferenceServiceConfiguration">
   <property name="list">
     <list>
         <bean class="org.dspace.app.cris.configuration.RelationPreferenceConfiguration">
                 <property name="relationConfiguration"  ref="relationRPVSItemConfiguration"/>
                 <property name="selectActionAuthLevel">
                         <util:constant static-
field="org.dspace.app.cris.configuration.RelationPreferenceConfiguration.ENABLED" />
                 </property>
                 <property name="hideActionAuthLevel">
                         <util:constant static-
field="org.dspace.app.cris.configuration.RelationPreferenceConfiguration.ENABLED" />
                 </property>
                 <property name="unlinkActionAuthLevel">
                         <util:constant static-
field="org.dspace.app.cris.configuration.RelationPreferenceConfiguration.ONLY_SYSTEM_ADMIN" />
                 </property>
                 <property name="extraAction">
                         <bean class="org.dspace.app.cris.configuration.ItemExtraAction" />
                 </property>
                 <property name="columnsVisualizationConfiguration">
                         <list>
                         <bean class="org.dspace.app.cris.configuration.ColumnVisualizationConfiguration">
                                 <property name="name" value="title" />
                                 <property name="metadata">
                                         <list>
                                                 <value>dc.title</value>
                                         </list>
                                 </property>
                         </bean>
                         <bean class="org.dspace.app.cris.configuration.ColumnVisualizationConfiguration">
                                 ...
                         </bean>
                         </list>
                 </property>
         </bean>
     </list>
   </property>
</bean>
```

The RelationPreferenceServiceConfiguration keeps a list of RelationPreferenceConfiguration that defines for a specific relationConfiguration which actions (select, hide, unhide, unlink[4]) are available and to whom (researcher and/or system administrators). Finally, the columnsVisualizationConfiguration allows to define which columns show in the relation management functionality (see) and the extraAction allows to inject the business logic needed to manage the preference changes (from a standard relationship to a selected one, from an hide relationship to a standard one and so on).

RELATION CLUSTERING

The configuration of the clustering categories are done using spring beans. The snippet below show the default configuration for a "journal article" category applied to the publication list

```
<bean id="journalComponent" class="org.dspace.app.webui.cris.components.BeanFacetComponent">
        <property name="componentIdentifier" value="journal" />
        <property name="facetQuery">
                <value>dc.type:Article -relationpreferences.crisrp.publications.selected:{1}</value>
        </property>
        <property name="order"><value>asc</value></property>
        <property name="rpp"><value>20</value></property>
        <property name="sortby"><value>1</value></property>
</bean>
```

The facetQuery property is evaluated replacing {0} with the authority key of current object and {1} with the uuid.

Statistics configuration

The SOLR Statistics system of DSpace has been extended in several aspects. It is now able to collect and visualize statistics about the CRIS Entities as allow further extension using StatsIndexPlugin in a similar way than the IndexerPlugin for the Search system.

The extended implementation is configured in the **[installDir]/config/spring/statistics-solr.xml** file

```
<bean class="org.dspace.app.cris.statistics.CrisSolrLogger" id="org.dspace.statistics.SolrLogger">
        <property name="spiderDetector" ref="org.dspace.statistics.util.SpiderDetector"/>
</bean>
<bean class="org.dspace.statistics.util.SpiderDetector" id="org.dspace.statistics.util.SpiderDetector"/>
```

The **[installDir]/config/spring/statistics.xml** file contains the spring beans definition to configure the extra indexers to invoke collecting statistics and most of the UI aspects of the statistics functionalities (number of "top relation", "top country" to show, timeline, etc.)

---

[4] The unlink action will remove the reference to the target object, i.e. unlinking an item from a researcherpage will remove the authority key in the corresponding item metadata

```
<bean id="identifierExtraIndexer" class="org.dspace.statistics.IdentifierStatsIndexPlugin"/>
```

Allow the indexing of the unique key of the SOLR search document in the statistics document so to allow the use of SOLR join query to retrieve statistics of related objects (i.e. number of visualization over time of all researcher's items, top visualized items for researcher, etc.)

The configuration of the statistics functionalities for the DSpace standard entities (Items, Collections and Communities) are done through spring beans. The following snippet show the default configuration for the DSpace Items

```
<bean id="itemStatsComponent" class="org.dspace.app.cris.integration.statistics.StatComponentsService">
        <property name="excludeBot" value="true"/> <!-- <value>-isBot:true</value>  -->
        <property name="showSelectedObject" value="true"/>
        <property name="topRelation" value="20"/>
        <property name="components">
                <map>
                        <entry key="selectedObject" value-ref="statsitem" />
                        <entry key="bitstream" value-ref="statsbitstreamforitem" />
                </map>
        </property>
        <property name="topCountryLength" value="10" />
        <property name="topContinentLength" value="-1" />
        <property name="topCityLength" value="10" />
        <property name="showMoreLength" value="20" />
        <property name="yearsQuery" value="5" />
</bean>
```

Most of the properties are self-explained; the one that need further details is components. A map of referring beans that describe the sections available in the statistics UI for DSpace Items.
There are two sections:
- selectedObject, is the reserved word to define the StatComponent "section" used to visualize the statistics of the current object
- bitstream, it defines a "section" where the top bitstream visualization are shown

```
<bean id="statsitem"
            class="org.dspace.app.webui.cris.components.statistics.StatSelectedObjectComponent">
        <property name="relationObjectType" value="2" />
</bean>

<bean id="statsbitstreamforitem"
```

```
        class="org.dspace.app.webui.cris.components.statistics.StatBitstreamTopObjectComponent">
        <property name="relationObjectType" value="0" />
        <property name="bean" ref="commonItemBean" />
        <property name="crisSearchService" ref="org.dspace.discovery.SearchService" />
</bean>

<bean id="commonItemBean" class="org.dspace.app.webui.cris.components.BeanComponent">
        <property name="componentIdentifier" value="collection" />
        <property name="query"><value>search.resourceid:{0}</value></property>
</bean>
```

Also the configuration of the statistics functionalities for the CRIS entities (ResearcherPage, OrgUnit, Project and DynamicObject) are done through spring beans. The following snippet show the default configuration for the ResearcherPages

```
<bean id="rpStatsComponent"
                class="org.dspace.app.cris.integration.statistics.CrisStatComponentsService">
        <property name="excludeBot" value="true"/> <!-- <value>-isBot:true</value>  -->
        <property name="showSelectedObject" value="true"/>
        <property name="showExtraTab" value="true"/>
        <property name="topRelation" value="20"/>
        <property name="topCountryLength" value="10" />
        <property name="topContinentLength" value="-1" />
        <property name="topCityLength" value="10" />
        <property name="showMoreLength" value="20" />
        <property name="yearsQuery" value="5" />
        <property name="crisComponentsService" ref="rpComponentsService"/>
        <property name="selectedObject" ref="rpSelectedObject"/>
</bean>
```

The bean definition is very similar to the definition used for standard DSpace Entities except for the components management. For the CRIS Entities the components are derived by the definitions done in the CrisComponentService, see "CRIS Components".
This mean that the user defined categories for clustering the relationship, for example the researcher's publication list by publication type, are used also in the statistics functionality so to show view and download of all the researcher's publication over time by publication type.
The current object is specified as direct property of the CrisStatComponent bean

```
<bean id="rpSelectedObject"
        class="org.dspace.app.webui.cris.components.statistics.CrisRPStatsDualSelectedComponent">
```

```
        <property name="applicationService" ref="applicationService" />
</bean>
```

These settings are injected in the UI using spring beans definition in **[webapp-JSPUI]/WEB-INF/springmvc-rp-servlet.xml**, the following snippet show the injection of the rpStatsComponets

```
<bean id="rpStatisticsController"
            class="org.dspace.app.webui.cris.controller.statistics.CrisStatisticsController">
      <property name="solrConfig">
            <ref bean="solrConfig" />
      </property>
      <property name="statsComponentsService">
            <ref bean="rpStatisticRequestsManager" />
      </property>
      ...
```

The rpStatisticRequestsManager is indeed defined in

[webapp-JSPUI]/WEB-INF/spring/applicationContext-rp.xml

```
<bean id="rpStatisticRequestsManager" factory-bean="researcher"
                              factory-method="getRPStatsComponents" />
```

where the factory bean researcher is only used to access the upper level "dspace-api" Spring Context.

Network configuration
The system is able to draw graph about network collaborations, see "Network visualization", between Researchers using data in the RPs, Projects, OrgUnits, Dynamic Objects and DSpace Items.

> ⚠ This functionality can be extensively configured editing a dedicated configuration file **[installDir]/config/modules/network.cfg**. Any property, plugin or setting listed below, if not otherwise stated, must be configured in that file

The system use the Jit javascript library (http://thejit.org/) to draw the graphs. A customized version of the RGraph layout has been developed. This customization changes the positioning algorithm of the nodes. Nodes are equally positioned in the "circle" corresponding to the degree of the relationship and not limited to the area

CUSTOMIZE

of circle "assigned" to the intermediate node at the previous level. Below there is a comparison between the DSpace-CRIS customization (on the left) and the standard RGraph layout.
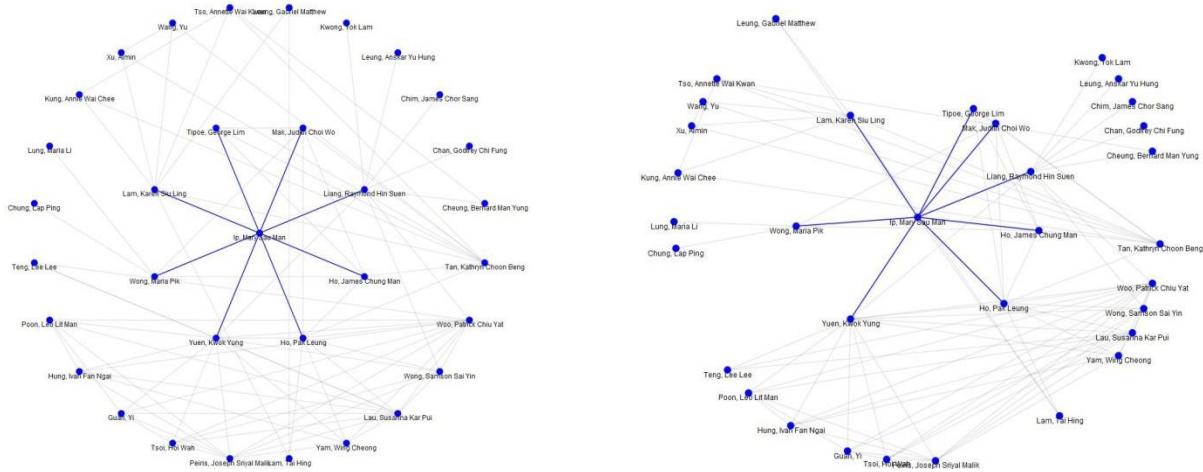
The relationships that are used by the system to build the network are defined as plugin in the **network.cfg** file

```
plugin.named.org.dspace.app.cris.network.NetworkPlugin = \
        org.dspace.app.cris.network.VisualizationGraphCoauthors = coauthors,\
        org.dspace.app.cris.network.VisualizationGraphInterests = interests,\
        org.dspace.app.cris.network.VisualizationGraphKeywordsPublication = kwdpub,\
        org.dspace.app.cris.network.VisualizationGraphCoinvestigators = coinvestigators,\
        org.dspace.app.cris.network.VisualizationGraphKeywordsGrants = keywordsgrants
```

all the connection-type needs to implement the it.cilea.surplus.oa.network.NetworkPlugin
A special, single plugin implementation is used to provide the "Department network: visualization & metrics"

```
plugin.single.org.dspace.app.cris.network.NetworkPlugin = \
        org.dspace.app.cris.network.DepartmentNetworkPlugin
```

The network.connection property define the exact order used to list the network in the UI

```
network.connection = coauthors, interests, coinvestigators, keywordsgrants, kwdpub
```

The UI load the graph data asynchronous via JSON. In the JSP UI there is a single servlet able to produce JSON for different purpose using plugins, these plugins are configured in the **dspace.cfg** file with the following names: collaborationnetwork, departmentnetwork

```
plugin.named.org.dspace.app.webui.json.JSONRequest = \
        ...
        org.dspace.app.webui.cris.servlet.JSONResearcherNetworkServlet = collaborationnetwork,\
        org.dspace.app.webui.cris.servlet.JSONDeptNetworkServlet = departmentnetwork
```

The colors used by a specific connection-type is defined as follow
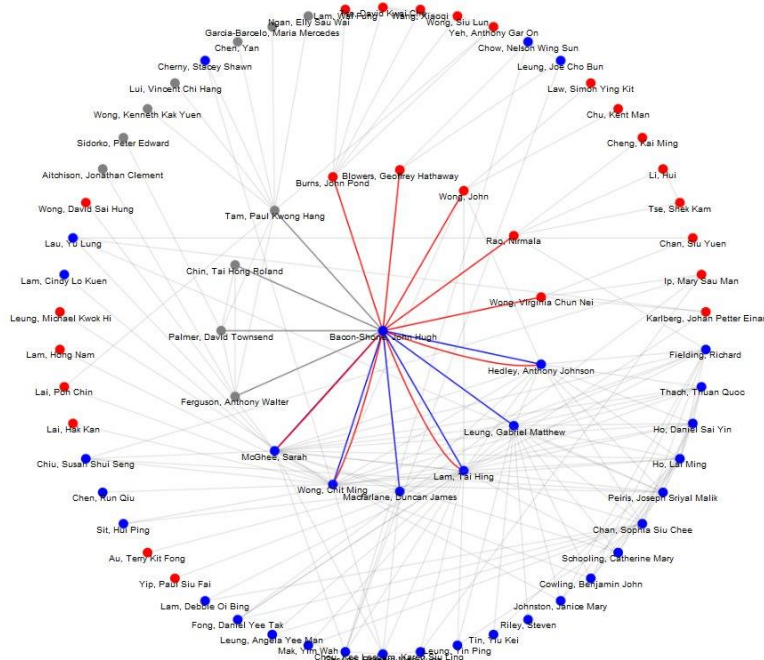
```
# node color

network.connection.color.node.default = #ddeeff

network.connection.color.node.interests = orange

# connection edge color

network.connection.color.edge.default = #C17878

network.connection.color.edge.interests = orange

# multi-connection edge color (only apply from the 2nd degree)

network.connection.color.edgetooverride.default = #C17878
```

As the number of nodes and connections in the graph grow, the graph will become more and more unreadable and less useful so the system allows to set some configuration properties that limiting the visualized data to the most important and will keep the graph more clear and fast to load.

```
# Max degree or relationship to show

network.connection.maxdepth = 2

# default value for the Show external people (not RP)
# (RP Network graph & Department graph)

network.connection.showexternal = true

# default value for the show only relationships inside the department
# (Department graph)

network.connection.showsamedept = false

# max number of nodes to show, for any node in the previous level,
# on each level (degree) of relationship (i.e. the first X co-author at the
# first level, the first X co-author of any first degree coauthor at the
# second level and so on. This mean that at the ith degree the graph could
# have a max of Xi nodes

network.connection.nodestoshoweachlevel.default = 30

# you can override all the previous settings on a connection-type/level
```

```
# basis

#network.connection.coauthors.nodestoshoweachlevel.0 = 15

#network.connection.coauthors.nodestoshoweachlevel.1 = 10

#network.connection.interests.nodestoshoweachlevel.0 = 15
```

Limiting the number of nodes shown could produce some unexpected behavior. For example, if you limit the display to the first 10 top co-authors than, in the case that, the 11th co-author of the focus RP is also in the top ten list of co-authors for one or more 1st level co-author this 11th co-author will be show as 2nd level relationship. But, when the user click on it, the system will report that the relationship with the focus RP is a first level relationship and the strength of the relationship will be shown.

For some relationship it is useful introduce some limit also during the indexing phase for both performance and meaning reasons. Looking for co-keywords relationship including all the keywords used in any publication is a no sense because some keywords are really general and widely used. Look to these keywords will produce a network of all the RPs and external authors in the repository. A better strategy is to only assign a keyword to an RP or external author if these people have used the keyword more than other. We look for all the keywords used in the repository and we assign any keyword to the top X authors that use it.

```
# -1 works on all possible results, configurable on a relationship basis

network.connection.loader.limitnode.default = -1

network.connection.loader.limitnode.kwdpub = 50
```

This limitation could be applied to the co-interest, co-discipline, co-keywords in grant, etc. as well.

The batch script

org.dspace.app.cris.batch.ScriptIndexNetwork (you can use the –a option to elaborate any connection-type or –s <alias> to elaborate a single connection-type) is used to build a dedicated SOLR core where network information are pre-assembled.

Network metrics

For any network the system is able to calculate four metrics
- Numbers of collaboration
- Max strength of the collaboration in the network
- Average strength of the collaboration network
- Quadratic variance of the collaboration network

These metrics are calculated by the batch script:

```
org.dspace.app.cris.batch.ScriptMetricsNetwork –a [or –s <alias> for single
connection mode]
```

These values are stored in the RP data so that, if desired, you can also show them in the public RP visualization

> ⚠ This feature requires that the following TextField exist in your RP configuration for each connection-type:

numberscollaboration_network_**<connection-alias>;**

maxstrengthcollaboration_network_**<connection-alias>;**

averagestrengthcollaboration_network_**<connection-alias>;**

quadraticvariancecollaboration_network_**<connection-alias>**

⚠ If you run the batch script without make the required changes to the configuration the script will prompt the missing field

The SOLR network core

The SOLR core used by the network visualization is defined as follow

```
network.server = http://localhost:8080/solr/network
```

In the network core a SOLR document represent a connection between two nodes (RP, external or department), the document includes the following fields:

- connection-type: the network alias (coauthors, interests, etc.)
- node A: text value, authority (rp key), rp department
- node B: text value, authority (rp key), rp department
- connection-value: the publication in common (co-authorship), the keyword (co-keywords), the interest (co-interest) etc.
- focus: these fields (values, authority) are automatically copied from the node fields to get fast response to some queries
- dept: this field contains the concatenation of the node A department and node B department field separated by a |||. It is used to get fast response to some queries

Use DSpace-CRIS entities as DSpace authority

All the DSpace-CRIS Entities can be used as authority for Item's metadata. The system provides the following ChoiseAuthority plugins

```
##### Authority Control Settings #####
plugin.named.org.dspace.content.authority.ChoiceAuthority = \
        org.dspace.app.cris.integration.RPAuthority = RPAuthority,\
        org.dspace.app.cris.integration.ProjectAuthority = ProjectAuthority,\
        org.dspace.app.cris.integration.OUAuthority = OUAuthority,\
        org.dspace.app.cris.integration.DOAuthority = DOAuthority
```

the following snippet from the default configuration show how to configure the ResearcherPages as authority for the dc.contributor.author metadata, the Projects as authority for the dc.relation metadata, the OrgUnits as authority for the dc.description.sponsorship and a DynamicObject "Journals" as authority for the dc.relation.ispartof metadata.

```
choices.plugin.dc.contributor.author = RPAuthority
choices.presentation.dc.contributor.author = suggest
authority.controlled.dc.contributor.author = true

choices.plugin.dc.relation = ProjectAuthority
choices.presentation.dc.relation = suggest
authority.controlled.dc.relation = true

choices.plugin.dc.description.sponsorship = OUAuthority
choices.presentation.dc.description.sponsorship = suggest
authority.controlled.dc.description.sponsorship = true

choices.plugin.dc.relation.ispartof = JOURNALAuthority
choices.presentation.dc.relation.ispartof = suggest
authority.controlled.dc.relation.ispartof = true
cris.DOAuthority.dc_relation_ispartof.filter = resourcetype_filter:"Journals"
```

It must be noted that the DOAuthority allows to define a SOLR query filter to limit the set of objects available as authority for example to only a specific Dynamic Object type.

RESOURCETYPE_FILTER is a SOLR field that contains the human-readable translation of the resource type (item, researcher, orgunit, etc.) it is configured in the **[installDir/config/modules/cris.cfg]** file as follow

```
### SOLR TYPE FACET – NOTE: separator \n|||\n must match the discovery configuration ###
facet.type.item = items\n|||\nItems
facet.type.community = communities\n|||\nCommunities
facet.type.collection = collections\n|||\nCollections
facet.type.crisrp = researchers\n|||\nResearchers
facet.type.crispj = projects\n|||\nProjects
facet.type.crisou = orgunits\n|||\nOrgUnits
facet.type.nested = nobjects\n|||\nN-Object

## You can define a clustered type for other research object or set a fine granularity configuration
## custom
## use @label@ to use default behaviour (<typedefinitionlabellowercase>|||<typedefinitionlabel>)
# facet.type.crisdo.default = others\n|||\nOthers
# facet.type.crisdo.crisprize = prize\n|||\nHonours, Awards and Prize
# facet.type.crisdo.crisqualification = @label@
facet.type.crisdo.crisjournals = @label@
```

If you have used the ItemEnhancer to add virtual metadata to the DSpace Item exploring the relationship with CRIS objects for the search and/or the browse system, see "ItemEnhancer: virtual metadata", you need to define an authority also for that virtual metadata to allow proper indexing of the authority key, variants and label.

```
## It comes from ItemEnhancer configuration - see cris-metadata.enhancers.xml
choices.plugin.crisitem.author.dept = OUAuthority
authority.controlled.crisitem.author.dept = true
authority.management.crisitem.author.dept = false
```

Chapter

4

SOAP WebServices

Database reference

The following E-R diagram shows the SOAP webservice configuration tables that allow to access to DSpace-CRIS data: accounts, access methods, filters, etc.

cris_ws_user	
🔑 id	INTEGER
enabled	BOOLEAN
password	CHARACTER VARYING(255)
username	CHARACTER VARYING(255)
showhiddenmetadata	BOOLEAN
fromip	CHARACTER VARYING(255)
toip	CHARACTER VARYING(255)
token	CHARACTER VARYING(255)
timestampcreated	TIMESTAMP(6) WITHOUT TIME ZONE
timestamplastmodified	TIMESTAMP(6) WITHOUT TIME ZONE
typedef	CHARACTER VARYING(255)

cris_ws_user2crit	
cris_ws_user_id	INTEGER
criteria_id	INTEGER

cris_ws_criteria	
🔑 id	INTEGER
criteria	CHARACTER VARYING(255)
enabled	BOOLEAN
filter	CHARACTER VARYING(255)

Powered by yFiles

Architecture

SOAP Request → Authentication → Authorization → Spring-WS

SOAP Response

Sender

Receiver
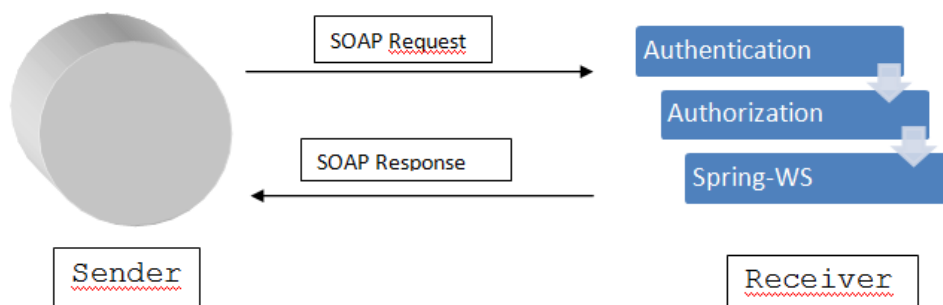
The "Authentication" and "Authorization" customized modules allow the administrator to manage access information.

Spring Web Services (Spring-WS) is provided by Spring community that is focused on creating document-driven Web services[5].

WSDL

A WSDL file is an XML document that describes a Web service. It specifies the location of the service and the operations (or methods) the service exposes. When creating Web services, there are two development styles: Contract Last and Contract First.

It was used Contract First because in Spring-WS it allows to start with WSDL definitions followed by Java code. Note that in Spring-WS, writing the WSDL by hand is not required. Based on the XSD and some conventions, Spring-WS may create the WSDL by itself.

Fragment of **spring-ws-servlet.xml:**

```
<bean id="wsdlforwsservices" class="…DynamicWsdl11Definition">

    <property name="builder">

     <bean class="…XsdBasedSoap11Wsdl4jDefinitionBuilder">
                <property name="schema"
value="<WEBINF>/crisrequestforwsdl.xsd" />

         <property name="portTypeName" value="cris" />

         <property name="locationUri" value="${dspace.url}/webservices/"
/>

         <property name="targetNamespace"

                         value="http://www.cilea.it/cris/definitions" />

    </bean>

    </property>

</bean>
```

Point browser or a SOAP client like SoapUI (download from here) to http://<dspace-webapp>/webservices/wsdlforwsservices.wsdl   Spring-WS can generate WSDL for us based on XML Schema called **crisrequestforwsdl.xsd**.

Others XML Schema imported on XML Schema for WSDL called *requestresearcherpage.xsd* and *responseresearcherpage.xsd*, *requestresearchergrants.xsd* and *responseresearchergrant.xsd*, *staticrequestpublications.xsd* and *staticresponsepublicatins.xsd* are not only your service contract but also are your data contract.

Data contract (with **request** prefix on xsd file name) defines the message format to accept and the objects type to return (with **response** prefix).

In fact, at begin of WSDL, you can find import of those xsd, this is the contract for request and response. Each *simpleType* of request contract defines object and field types to respond.

There are two schema related to WSDL:

- **crisrequestforwsdl.xsd** in the classpath is used to generate public WSDL;
- **crisrequest.xsd** in the external folder is used to validate data request.

Note that **crisrequestforwsdl.xsd** and **crisrequest.xsd** have the same contents, they differ only by the initial imports.

```
simpleType name="ValueListPublications">

        <xs:restriction base="xs:string">

                <xs:enumeration value="search.resourceid" />

                <xs:enumeration value="dc.type" />

                <xs:enumeration value="dc.title" />

                ...

                <xs:enumeration value="dc.contributor.author" />

        </xs:restriction>

    </xs:simpleType>
```

Validation is done on request call based on xsd. Note that for entities in the system the xsd like for RP entity or GRANT entity will generate at system startup and put them in **ws-xsd** (configurated on dspace.cfg with key *webservices.xsd.path*) directory on dspace local directory. The only one to copy onto directory by hand is request contract and response contract for publications (you can find these in **WEB-INF\classes** directory onto webapp).
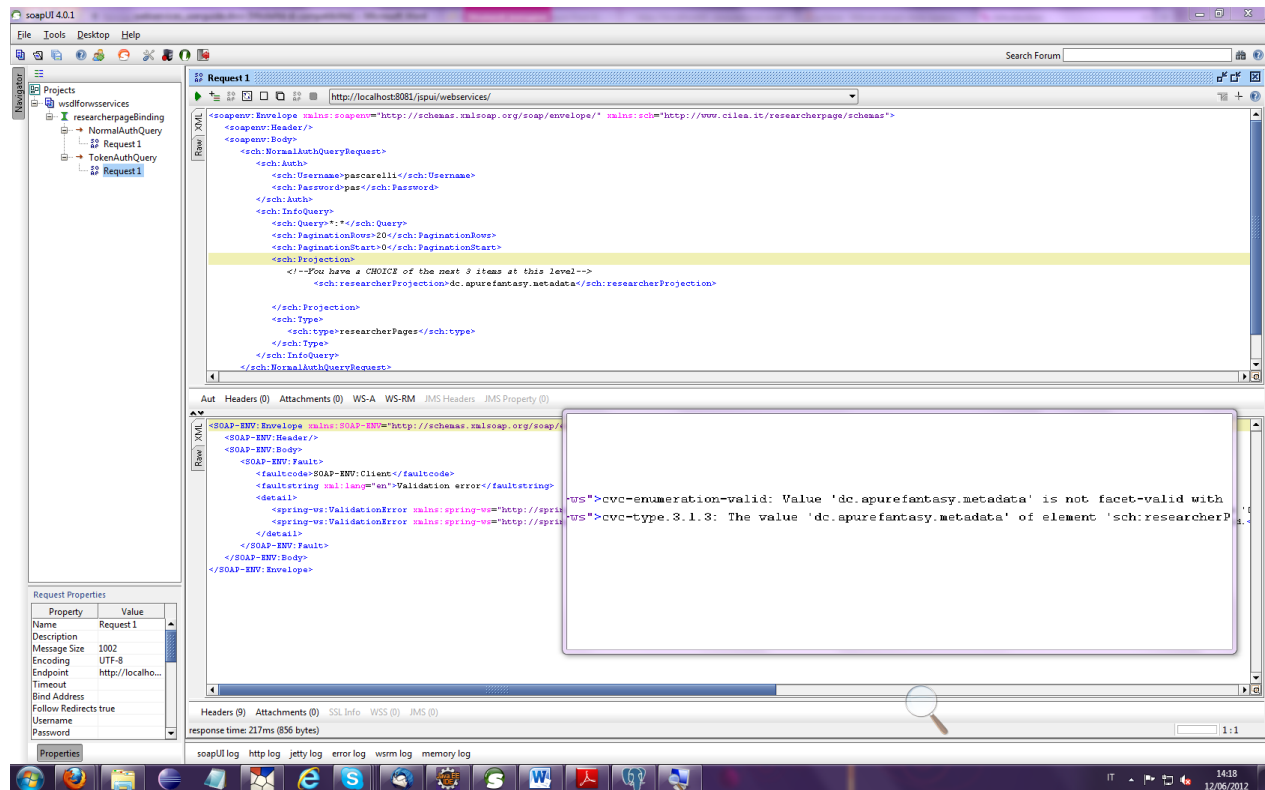
The service endpoints are managed by **PayloadRootQNameEndpointMapping** class provided by Spring Framework.

This class maps SOAP request

"{http://www.cilea.it/cris/schemas}NormalAuthQueryRequest"

to

rpNormalEndpoint

bean service and

"{http://www.cilea.it/cris/schemas}TokenAuthQueryRequest"

to **rpTokenEndpoint** bean service.

rpNormalEndpoint and rpTokenEndpoint first authenticate and authorize user, then manage request and build response to fulfill data contract.

Web Service User Management

The administration menu shows the "Web services" item through which the web service users may be managed.

It allows to create two kind of user authentication:

- username/password
- with a token and a range or single IP

It also allows to:

- enable or disable users
- enable or disable permissions

- filter criteria (SOLR-like)
- show values

Objects managed appear in dspace.cfg:

```
##Web Services Configuration

webservices.criteria.type = publications, researcherPages, grants

webservices.criteria.type.publications = 2

webservices.criteria.type.researcherPages  = 9

webservices.criteria.type.grants  = 10
```

Note: to add new entity it is mandatory add new type in dspace.cfg and also manage XML Schema definition.

Web Service Test
SOAP UI allow to contact web services. Followed it is the SOAP XML document aka SOAP envelope. For further information see the code fragment comments.
Follows a token authentication request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:sch="http://www.cilea.it/researcherpage/schemas">

   <soapenv:Header/>

   <soapenv:Body>

      <sch:TokenAuthQueryRequest>

         <sch:Auth>

             <!-- the token -->

            <sch:Token>xsbcsdsdfsf-awe12-asdad</sch:Token>

         </sch:Auth>

         <sch:InfoQuery>

             <!-- the query -->

            <sch:Query>*:*</sch:Query>

             <!-- pagination -->

            <sch:PaginationRows>20</sch:PaginationRows>

            <sch:PaginationStart>0</sch:PaginationStart>

            <sch:Projection>

               <!--You have a CHOICE of the next 3 items at this level-->

               <!-- CHOOSE only one and take the consistent type -->

               <!—if leave empty then the result depends on user profile->

               <sch:grantProjection>search.resourceid
```

```
grant_keywords</sch:grantProjection>

            </sch:Projection>

            <sch:Type>

                    <!-- publication,researcherPages,grants are allowed -->

              <sch:type>grants</sch:type>

            </sch:Type>

        </sch:InfoQuery>

    </sch:TokenAuthQueryRequest>

  </soapenv:Body>

</soapenv:Envelope>
```

Server side, the token, get by SOAP messages, and the caller address IP, get by transport service aka HTTP, are validated. If validation is true then a query SOLR is built in the follow manner:

- Getting query from SOAP message
- Getting type and building a query (es. fq=search.resourcetype:2 where 2 stands for Item, 9 for RP and 10 for Grants)
- Getting in the from User profile other queries
- Shoot query on SOLR server (reminder: pagination is active)

Follows a sample response:

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">

   <SOAP-ENV:Header/>

   <SOAP-ENV:Body>

      <cris:TokenAuthQueryResponse hit="15206" rows="1" start="0"
type="grants" xmlns:cris="http://www.cilea.it/cris/schemas">

         <grant:crisobjects
xmlns:grant="http://www.cilea.it/grant/schemas">

            <grant:crisobject businessID="4" publicID="1" type="10"
uuid="f5c01ae7-0921-4b9b-be84-c66bacaa6011
">

                <grant:fundingyear
```

```
visibility="1">1995/1996</grant:fundingyear>

                <grant:amount visibility="1">12000</grant:amount>

                <grant:startdate visibility="1">01-08-1995</grant:startdate>

                <grant:extdate visibility="1">31-07-1996</grant:extdate>

                <grant:projecttitle visibility="1">Biomechanical evaluation
of the human sacrum for lumbosacral fixation</grant:projecttitle>

                <grant:keywords visibility="1">orthopaedic</grant:keywords>

                <grant:discipline
visibility="1">Orthopaedics/Traumatology</grant:discipline>

                <grant:granttype visibility="1">Departmental Budget -
General Award</grant:granttype>

                <grant:status visibility="1">Completed</grant:status>

                <grant:investigator visibility="1">Professor Leong John Chi
Yan</grant:investigator>

                <grant:coInvestigators>

                    <grant:coInvestigator rpkey="411">Lu, William
Weijia</grant:coInvestigator>

                    <grant:coInvestigator>Professor
Zhong</grant:coInvestigator>

                    <grant:coInvestigator>Dr Zhu</grant:coInvestigator>

                </grant:coInvestigators>

            </grant:crisobject>

        </grant:crisobjects>

    </cris:TokenAuthQueryResponse>

  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

The administrative UI

A new menu item "CRIS Module" has been added to the DSpace administrative menu list. In this area is possible to manage the objects introduced by the CRIS module: Researchers, Projects, OrgUnits and DynamicObjects. For any entity is possible to manage the data structure: fields and relationship with other objects and the presentation aspects. In the following paragraphs we will use the ResearcherPage entity as reference but the same concepts apply in the same ways to all the other entities.

The "ResearcherPages" option of DSpace Administration menu presents the following functionalities, that allow to set up layout and content of the Researcher Pages:

- Layout configuration (Public view)

- Layout configuration (Edit view)

- Layout configuration (Group metadata)

- Add Researcher page

- View all Researcher

Lay out configuration (Public view / edit view)

The "public" and "edit view" of the "Layout configuration" function refer to the same concept. The first one allows to organize the Researcher data across one or more Tabs that are publicly accessible while the second one organize the Researcher data across one or more edit screen (tabs).

The following figure shows a Researcher's Personal Page.

At the top of the page there are two tabs: "Personal Page" and "Cerif". Furthermore, each tab contains one or more collapsible boxes, each one containing the researcher's data.

The following figure shows the configuration side of the Researcher Page previously viewed.

"Personal Page" and "Cerif" tabs may be edited or deleted.

⚠   Deleted tab cannot be restored, it has to be created again.

It is also possible to create new tab clicking on the "CREATE NEW TAB" link, at the bottom of the page

Tab Configuration
By clicking on "edit" option, the system displays the following form.

SHORTNAME: it represents the unique identifier of the record (if you want to modify it your installation manager must be informed)

TITLE: name of the tab displayed in the ResearcherPage.

PRIORITY: numeric value that indicates the priority of the displayed tabs (if two or more tabs have the same 'priority' value, the labels will be displayed as before).

VISIBILITY: it indicates who can access to the tab. The options are:

- High level (all): accessible to anyone

- Show to RP owner and to admin: accessible only by the page owner and by the administrator

- Show only admin: accessible only by the administrator

- Show only to RP owner: accessible only by the page owner

The "TAB ICON" section allows to upload a file in order to set the tab icon.

The "BOXES" section shows the list of all existing boxes. Tick boxes are included in the current Tab (see also "Group Metadata, aka boxes" for further details).

Group Metadata, aka boxes

A box is a group of metadata that are shown/edit together. In this section is possible to see all the boxes configured in the system, edit the current configuration for any box or add a new one.
For each one, the system other than the usual edit and remove icons can show two special "JSP icon":

- A yellow "JSP icon" appears near the box name, it means that a JSP fragment does exist in order to set up the box for the public display;

- A red "JSP icon" is present when the fragment is valid for "edit view.

Custom JSPs are normally used when you want to override the out-of-box layout produced by the system, see "Custom JSPs" for further information.

By clicking on the "edit" option of a box with JSP icon, the system shows the following mask:

The **EXTERNAL RENDERING** field contains the name of the custom JSP page, if any, to use for the box rendering. In the example a rpcerif**.jsp** file needs to be placed in the **dspace-cris/jdyna/custom** folder of the webapp.

If **SHOW COLLAPSED** is flagged, the box will be shown collapsed in the public view and the user will need to click the box title to see the content.

The **UNNECESSARY** flag exclude the box from the evaluation of the Tab content, if a Tab has not content the tab is hidden.

In the **FIELDS AVAILABLE** container are shown all the already defined fields. Ticked fields are included in the box. By default, only the fields in the box are shown, to see the other available fields you need to click on the Show/Hide other fields link.

Data Model configuration: Field Definition

From the edit/create box form is possible to create new fields.

The following image shows the "edit" option of a box configuration. The field definition menu is placed at the bottom of the page (look at the red circle in the figure).

The available data type are: **DATE**, **TEXT**, **COMPLEX**, **LINK** and **FILE** other than reference to other CRIS objects.

The basic set of metadata that define any type of field is enclosed into the upper red frame and this set includes: Shortname, Label, Field access level, Mandatory, Repeatable, Priority, New Line, Label min size and Field min size for row and column.

A field is determined by the following metadata:
- a "Shortname" and a "Label" to identify the field
- a "Field access level" to define the access level for the field in the edit screens (NB: Administrators can always edit/hide any field)
- a "Mandatory" flag to make field required (no effect on property definition created by nesting)
- a "Repeatable" flag to give the possibility to add more objects of the same type
- a "Priority" index to increase or decrease the priority of the field, thus moving it up or down in the screen
- a "New Line" flag to require new line after the field
- a "Label min size" to define the minimum size of the label
- a "Row" and "Column" "Field min size" to define the minimum size of row and column where the field is positioned

DATE
Two extra elements are specific for the Date field:
- an "Year Min" and "Year Max" to set the calendar minimum and maximum year.

TEXT

Two extra elements are specific for the Text field:
- the "Size" metadata sets the textbox size
- the "Regex" metadata is available to validate a regular expression.

COMPLEX

It is a nested object made up of one or more data types. As the next figure shows, the user has the possibility to add every kind of element to be nested (a text, a date, a link, a file, a researcher profile pointer, a project pointer, an organization unit pointer, a DO pointer), each one with its specific form explained in this documentation.

The basic set of metadata in the upper red square of the complex type are the same of the previously mentioned fields, except for the last one named "Inline or tabulized": when it is checked it means that data are not shown inside a table, otherwise when unchecked data are shown in table format.

LINK

data type allows to create an hypertext field. There are two specific elements:
- head label, which is a header label that may be shown while editing the link
- head label URL, which is a header label that may be shown while editing the URL.

FILE

data type allows to upload a file, with the set of metadata shown in the next figure.

Basic metadata are the same as above, whereas specific metadata for the file data type are the following:

- Show preview: this element must be checked in order to show a preview of the file in the detailed page
- File description: the user has the possibility to write here a short description to use for upload
- Label hypertext: this is a hypertext link
- Keep track of downloads into statistics index: the user may decide to keep track of the number of downloads of the file to build up statistics on it.

Reference fields

It is also possible to create four other types of field:

- Add a Researcher Profile reference field
- Add a Project reference field
- Add an Organization Unit reference field
- Add generic research entities reference field

These functionalities allow to set up a relation between a researcher (assuming that we are managing researcher page fields) and another entity, via the form reported in the following figure:

Other than the basic list of metadata there are the following specific elements:

- Value to display (EL): it has to be expressed using the Expression Language (EL) syntax, e.g "${displayObject.fullName}" or "${displayObject.anagrafica4view['fullName'][0].value}".
- Filter query: any SOLR query that returns the appropriate data set must be typed here, in order to filter the lookup set.
- Index name: Name of the SOLR index to query on

- URL Path for resource: the relative URL to use to access the details of the linked object assuming as base URL the webapp application context path. The expression in the url path will be evaluated.
  For example: cris/uuid/${displayObject.uuid}

Entities management

The system allows the administrators to create, edit, remove and list instances for any defined CRIS Entity. The following paragraphs describe these functionalities for the ResearcherPage but apply as well to all the other entities included the Dynamic Objects.

Add an Entity instance (for example a ResearcherPage)

In order to add a new instance of a ResearcherPage, the system provides the "Add ResearcherPage" function.

"Profile" field indicates whether the entity is public or private. A private entity is not searchable and it is visible only by the administrator user.

View all the entity instances (for example all the Researchers)

The functionality allows to display the whole list of researchers stored in the system. By clicking on a researcher Internal ID or UUID, the system shows the related researcher page.

"Status" flag means public or private entity as well as "Profile" radio field in edit researcher page (see "AddResearcherPage" paragraph). If "Status" is checked the researcher is a public entity. Otherwise it is private (default).

Configure the CERIF Mapping

To be. It will be available in the final release not in the beta release.

The researcher UI

Create a personal profile

In the "My DSpace" section, Researchers are allowed to have a personal page inside DSpace-CRIS and this page is called "Researcher profile".

A Researcher may hide or remove his own researcher profile: when the profile is currently active, it can be hidden from the public view or removed completely from the system. Hidden profiles are still searchable in submission to link authored publications. The possibilities are listed in the following figure:
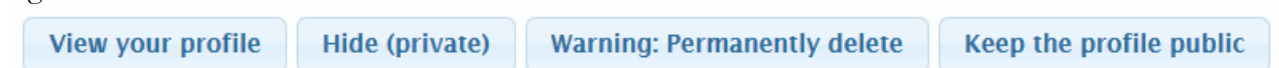
By clicking "View your profile" the user may access his personal page.

Edit personal profile

To modify data, the user must press the blue tab on the left, labelled as "Edit Researcher Page", to get the following view:

Then, expanding the two forms named "Details" and "Misc", all elements are shown:

If one element is checked, it will be shown in the public view, conversely if unchecked it will not appear.

When the label is red (such as "full name"), it means that the field is required; when the label is black, it is optional. . As far as affiliations are concerned, they must be inserted throw auto-compiling, that

means that it is enough to write three characters to start a search in the data base and find the desired organization unit.

Manage relations: selected publication

Publications on the Researcher's personal page can have one of three categories:

- Active publications, that are displayed normally,

- Hidden publications, that are not displayed in the ResearcherPage, but are still searchable in the Institutional Repository,

- Selected publications, that are moved in the "Selected Publications" box.

The "Selected Publications" box shows all the publications that have been selected, and allows to reorder them as whished using drag & drop.

For any particular publication, the user is able to:

- Change Category: For any publication entry, it is possible to change the category by clicking on the category icon A (active), H (hidden), S (selected), or U (disclaim)

- Filter: it is possible to use the filter to search through publications.

It is important to click "Submit" after making changes, not to lose the new configuration.

The list of selected publications, in the given order, appear in the personal page as shown in the following figure:

DSpace submission

Researchers upload their publications in the repository through the submission process.

At any point in the submission process the work can be stopped and saved for a later date by clicking on the "cancel/save" button at the bottom of the page. The data already entered will be stored until the user will come back, and he will be reminded in "My DSpace" page that there is a submission in process. It is also possible to cancel the submission at any point.

The submission process is divided in seven steps that are highlighted in the Progress Bar at the top of the page:

Six oval buttons represent each step of the submission process. As the user moves through the process these ovals will change color. Once started, it is also possible to use these buttons to move back and forth within the submission process by clicking on them without losing any data.

The first step is to select a collection: the user must click on the arrow at the right of the drop-down box to see a list of Collections, and then move the mouse on the name of the collection into which he wants to add the new item.

The researcher has to be authorized by a community to submit items to a collection. If you would like to submit an item to DSpace, but don't see an appropriate community, you should contact your DSpace Administrator to find out how you can get your community set up in DSpace.

The user must click on the "next" button to proceed, or click on the "cancel/save" button to stop and save or cancel submission.

SUBMIT: Describe Your Item - Page 1

Describe | Describe | Upload | Verify | License | Complete

Submit: Describe this Item

Please fill in the requested information about this submission below. In most browsers, you can use the tab key to move the cursor to the next input box or button, to save you having to use the mouse each time. (More Help...)

Enter the names of the authors of this item below.

Authors

de Castro, P. ✓ Remove This Entry

Bollini, A ✓

Add More

Enter the main title of the item.

Title Higher education

Please give the date of previous publication or public distribution below. You can leave out the day and/or month if they aren't applicable.

Date of Issue *Month:* February *Day:* 12 *Year:* 2010

Enter the name of the publisher of the previously issued instance of this item.

Publisher Springer

Select the type(s) of content of the item, get by CERIF Semantics

Type Journal Article ✓ Add More

Select the language of the main content of the item. If the language does not appear in the list below, please select 'Other'. If the content does not really have a language (for example, if it is a dataset or an image) please select 'N/A'.

Language Italian

Next >   Cancel/Save

SUBMIT: Describe Your Item - Page 2

The information filled in on these two screens will form the metadata record that will enable users to retrieve the item using search engines. The richer the metadata, the more "findable" this item will be, so it is important to take the time to fill in as many fields as are applicable to this item.

The author can be a person, organization or service responsible for creating or contributing to the content of the item. By clicking on the "Add More" button it is possible to add as many authors as needed. The user may choose among the list of authors that are saved in the institutional data base, or to insert a brand new name.

The title is the full and proper name by which this item should be known. All DSpace items MUST have a title.

If this item was previously published or made public, the date of that event may be entered in the "date of issue" field. If you don't know the month, leave the default "no month"; otherwise select a month from the drop-down box. If you don't know the exact day, leave that box empty.

The name of the publisher of this item may be entered in the "publisher" field.

The type of work that best fits this item can be selected. It is even possible to add more than one type or genre.

The language of the intellectual content of the item may be selected among a list of languages commonly used for publications. If the item is not a text document and language is not applicable as description, then the N/A choice may be selected.

To describe the item, researchers may enter as many subject keywords as are appropriate, from the general to the specific. The more words are provided, the more likely it is that users will find this item in their searches. One different input box must be used for each subject word or phrase. More input boxes are added by clicking on the "add more" button. The institution may suggest the use of a specific vocabulary, taxonomy, or thesaurus. If this is the case, researchers should select subject words from that list. Future versions of DSpace will provide links to those lists.

As far as abstract is concerned, it is possible either to cut and paste an abstract into this box, or to type in the abstract. There is no limit to the length of the abstract. Include an abstract is useful for the convenience of end-users and to enhance search and retrieval capabilities.

If the item is the product of sponsored research, information about the sponsor may be provided in the "sponsor" field. This is a freeform field where any note may be entered.

SUBMIT: Upload a File

Clicking on the "browse" button a window showing file system will appear. The user can navigate through directories and folders to find the correct file to upload. Double-clicking on the chosen file name, it will be entered into the input box.

There is also an input box marked as "File Description": the information provided here will help users to understand what information is in each file, for instance, "main article" or "images" or "computer program" or "data set".

Having uploaded a file, check the information in the table to make sure it is correct.

There are two further ways to verify that files have been uploaded correctly:

- Click on the filename. This will download the file in a new browser window, so that contents may be checked.

- Compare the file checksum displayed here with the calculated checksum.

To upload more than one file, the user clicks on the "Add Another File" button.

DSpace generates an MD5 checksum for every file it stores; we use this checksum internally to verify the integrity of files over time (a file's checksum shouldn't change). Researchers may use this checksum to be sure what DSpace has received is indeed the file uploaded.

SUBMIT: Verify Submission

This page lets the user to review the information he has entered to describe the item. To correct or edit information, click on the corresponding button on the right, or use the oval buttons in the progress bar at the top of the page to move around the submission pages.

SUBMIT: License

DSpace requires agreement to this non-exclusive distribution license before any item can appear on DSpace. Researchers should read the license carefully.

SUBMIT: Submission Complete

Now that the new submission has been successfully entered into the DSpace system, it will go through the workflow process designated for the chosen collection. Some collections require the submission to go through editing or review steps, while others may immediately accept the submission. The researcher will receive e-mail notification as soon as the item has become a part of the collection, or if for some reason there is a problem with the submission. It is possible to check on the status of the submission by going to the My DSpace page.

Chapter

7

The end user UI

Search & browse

Entity detail

DSpace-CRIS at euroCRIS >

Researcher profile: Bollini, Andrea

Tabs & boxes

Publication list clustering

Statistics

Publication list clustering

Network visualization

The system is able to draw graph about network collaborations between Researchers using data in the RPs, Projects, OrgUnits, Dynamic Objects and DSpace Items. This functionality can be extensively configured, see ''These settings are injected in the UI using spring beans definition in **[webapp-JSPUI]/WEB-INF/springmvc-rp-servlet.xml**, the following snippet show the injection of the rpStatsComponets

```
<bean id="rpStatisticsController"
        class="org.dspace.app.webui.cris.controller.statistics.CrisStatisticsController">
    <property name="solrConfig">
        <ref bean="solrConfig" />
```

```
        </property>
        <property name="statsComponentsService">
                <ref bean="rpStatisticRequestsManager" />
        </property>
        ...
```

The rpStatisticRequestsManager is indeed defined in

[webapp-JSPUI]/WEB-INF/spring/applicationContext-rp.xml

```
<bean id="rpStatisticRequestsManager" factory-bean="researcher"
                          factory-method="getRPStatsComponents" />
```

where the factory bean researcher is only used to access the upper level "dspace-api" Spring Context.

Network configuration".

The system use the Jit javascript library (http://thejit.org/) to draw the graphs. A customized version of the RGraph layout has been developed. This customization changes the positioning algorithm of the nodes. Nodes are equally positioned in the "circle" corresponding to the degree of the relationship and not limited to the area of circle "assigned" to the intermediate node at the previous level. Below there is a comparison between the DSpace-CRIS customization (on the left) and the standard RGraph layout.
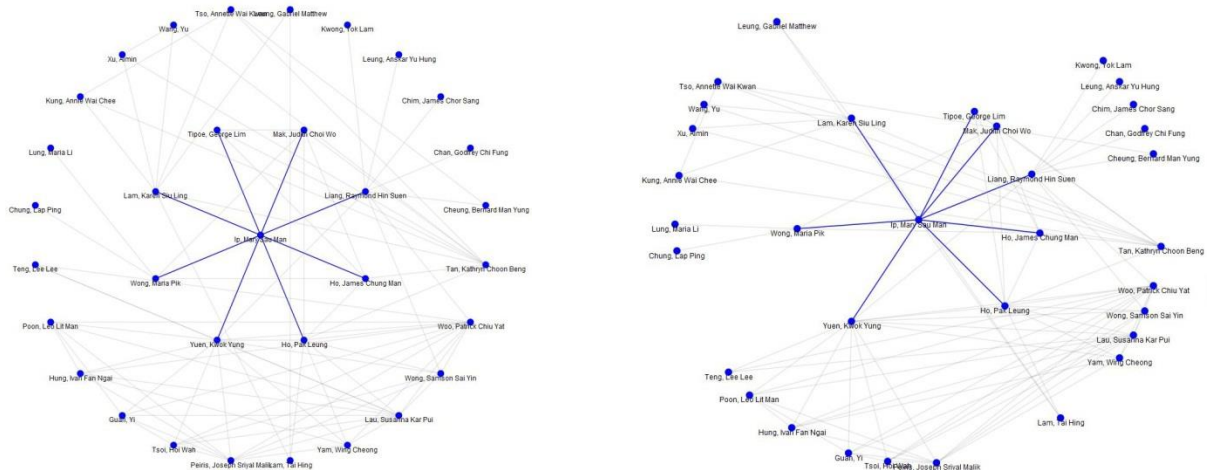
The user can choose the graph layout to use from the dropdown menu on the bottom left of the network visualization tool.

The Network configuration box as shown in the next screenshot allow the user to decide which relations use to build the graph: co-authorship, co-investigator, co-keywords, etc. and for each one how many degrees of connections to show.
The system shows only the network that have data for the current RP. Double-clicking on a relation name will automatically select this relation and deselect any other.

When a node appears in two of more relations, the node will get the color of the first relation that have shown it but at the first level all the connections edge are shown with different curvature so to give notice of the "multi-relationship" between the focus and the connected node.

Moving the mouse over a node will show a toolbox with the name card of the node and will highlight the path between the node and the focus. Clicking on a node will trig the load of additional information about the node and the connections between this node and the focus. If there are no direct connections between the node and the focus only the minimum degree of relationship is shown otherwise the strength of relationship for any connection-type is showed.

clicking a popup with the details of the connection is showed

As the number of nodes and connections in the graph will grow the graph will become more and more unreadable and less useful so the system allows to set some configuration properties that limiting the visualized data to the most important and will keep the graph more clear and fast to load.

Department network: visualization & metrics
The name card of the RPs give access to the department graph through the network icon

Department network focus the graph on a department and will show with the same relationship and criteria explained above for the single RP all the RPs in the department. From any RP in the department the graph will show relationship between any RPs, RPs in other department and external people using the option defined by the user (max depth, enabled connection-types, show external, show other department).
The department network will also include a box to highlight department metrics

Dept of Medicine

- External researchers connected: 2491
- RP from others departments: 275
- Others departments connected: 40
- RP connected: 401
Show metrics

and a link (show metrics) to the summary metrics of any RP in the department.

Credits

Andrea BOLLINI <a.bollini@cineca.it>
Chiara MOLINARO
Luigi Andrea PASCARELLI
Marta ZAETTA