configure the data model using an Excel file

The DSpace-CRIS data model configuration can be exported and imported using an excel file.

To export the current datamodel in excel you need to run

```
./dspace export-cris-configuration -f {fullpath-to-the-target-file.xls}
```

to import the datamodel

```
./dspace load-cris-configuration -f {fullpath-to-the-target-file.xls}
```

The import script will work in append mode so you can use it to initially populate your datamodel in an empty dspace-cris installation but also to add new attributes or entities.

It is not able to update the definition of existent attribute

File structure

The excel file is composed of several sheets

- **propertiesdefinition:** it contains the list of all the properties defined for all the entities
  - **column A (TARGET)**: it contains the shortname of the entity that holds the property. rp, ou, pj are the reserved words used for Researcher Pages, OrgUnits and Projects. Any other values must be a shortname assigned to a new entity defined in the datamodel (DynamicObject) and added to the **utilsdata sheet in the column C**.
  - **column B (SHORTNAME)**: it is the unique name assigned to the property. It is used in the import/export tools as header of the excel file and in all the configurations to refer to the property
  - **column C (LABEL)**: the Label shown to the end user in the public and/or edit pages
  - **column D (REPEATABLE)**: it defines if the property accepts multiple values y (yes) or, single value n (no)
  - **column E (PRIORITY)**: the properties are ordered using a descending order. Highest value put the property at the top of the box
  - **column F (HELP)**: an hint shown to the user in the edit page
  - **column G (ACCESS LEVEL)**: the security level of the property. It can be
    - *HIGH_ACCESS*: visible and editable by anyone able to access (in view and/or edit mode) the box where it is contained
    - *STANDARD_ACCESS*: only apply to RP properties. The RP owner can only set the visibility of the property values but not edit the content
    - *ADMIN_ACCESS*: only a system administrator or a CRIS entity administrator can view and edit the property
    - *LOW_ACCESS*: only apply to RP properties. The property is visible only to the RP Owner nor to the system administrator
  - **column H (MANDATORY)**: is the property required or no? if y it will prevent save in the edit if no filled
  - **column I (WIDGET)**: the type of values of that the property holds
    - nested
    - text
    - date
    - link
    - file
    - pointer
    - image
    - boolean
    - radio
    - checkbox
    - dropdown
    - eperson
    - group
    - classificationtree
  - **column J (POINTER CLASS)**: it should be n for all the widget types except pointer. If Widget = pointer it is the shortname of the linked entity (i.e. rp, ou, pj, etc.), instead if Widget = classificationtree then this cell need linked entity and related metadata (e.g. events|||eventsrelatedinterpretation)
  - **column K (RENDERING)**: only used for Widget = pointer. It contains the HTML fragment (EL supported) to use to visualize the linked Entity
  - subsequent columns are used to define the size and positioning of the property in the autogenerated layout
- **nesteddefinition**: it contains the list of nested properties for all the entities. A nested property is a structured information composed of several sub properties such as an Address (Street, City, CAP, etc.) or an affiliation (start/end date, role, institution, etc.)
  - **column A (TARGET)**: the target entity (rp, ou, ...)
  - **column B (SHORTNAME)**: the shortname of the sub property, similar to column B of propertiesdefinition
  - **column K (ANCESTOR)**: the shortname of the "whole" nestedobject, i.e. the address, affiliation etc. It must match a shortname of a row in the propertiesdefinition sheet with widget = nested
- **tab**: it contains the list of sections for each entities for the public pages
  - **Column F (VISIBILITY):**
    - *HIGH_ACCESS*: visible by anyone
    - *LOW_ACCESS*: only apply to RP tabs. The tab is visible only to the RP Owner nor to the system administrator
    - *POLICY_ACCESS*: access to the tab is allowed only to users and groups listed in another property (see tabpolicy)

- **etab**: it contains the list of sections for each entities as used in the edit mode
  - **Column F (VISIBILITY):**
    - *HIGH_ACCESS*: editable by anyone that has access to the edit mode
    - *LOW_ACCESS*: only apply to RP tabs. The tab is visible only to the RP Owner nor to the system administrator
    - *POLICY_ACCESS*: access to the edit tab is allowed only to users and groups listed in another property (see etabpolicy)
- **box:** it contains the panel that are used to aggregate properties in view and edit mode inside a tab/etab
  - **Column G "Unrelevant"**: it can contain y or n and it is used to determine at runtime if a tab need to be displayed for a specific object or not. A tab is displayed only when it contains actual data, ie there are at least one box that contains data in the tab. Only box flagged with unrelevant = n are checked. In this way, if you have a general box that is reused across multiple tabs to show basic information like a "name card" the tab is displayed only if there are information in some other boxes
  - **Column H (VISIBILITY):**
    - *HIGH_ACCESS*: editable by anyone that has access to the edit mode
    - *LOW_ACCESS*: only apply to RP boxes. The tab is visible only to the RP Owner nor to the system administrator
    - *POLICY_ACCESS*: access to the box is allowed only to users and groups listed in another property (see boxpolicy)
- **tab2box**: it maps a box to one or more (view) tab
- **etab2box**: it maps a box to one or more (edit) etab
- **box2metadata**: it maps a property to one or box
- **utilsdata:** It contains data used to validate the content of the other sheets. The only column that need to be edit is the column C that need to contains all the entities defined in the data model (rp, ou and pj are reserved words for the top level CRIS object Researcher page, OrgUnit and Project). See also the sheet **propertiesdefinition column A**
- **controlledlist**: it contains the list used for dropdown,checkbox and radio widget types. Each column contains the list of values available for propertydefinition identified by the column heading (= shortname). Each value must be in the form <stored-value>###<display>
- **tabpolicy**: it defines which properties are used to check the access condition of tab with **ACCESS LEVEL = POLICY_ACCESS.** Target is the shortname of the entity, shortname must match the tab shortname, metadata is the shortname of the property that will enforce the check and type can be eperson or group (it must match the widget type of the propertydefinition)
- **etabpolicy:** see above but for tab used in the edit mode
- **boxpolicy**: see above, for the security of the box