Technical Note

Extending the Rep++ Metadata with User-Defined Attributes

Overview

There are times when the metadata associated to a number of standard Rep++ objects (e.g. Field, Rowset, RowsetTree, etc.) is simply insufficient to describe a particular kind of data. For instance, there have been a number of projects where:

- For each field, we had to maintain a URL pointing to online documentation for that field. This URL was later used by a custom report generator to produce hyperlinks to the online documentation.
- For each field, we had to add a flag that indicated the background color for the field column in a grid.
- For certain fields, we had to maintain the name of a cache item that stored a list of choices available for the field, in order to avoid re-reading the list every time.
- For each Rowset, we had to specify the location of a custom report template that was used by a custom report generator.

To implement these capabilities, we used user-defined attributes (UDAs). A UDA is an element of information added by the developer to extend the metadata of a number of standard Rep++ objects in order to store additional information. UDAs can be associated to Fields, Rowsets, RowsetTrees, RowsetTreeDefNodes, etc. They are created in a special editor that lets you define their type, the object they are associated with, their size, and other related aspects.

This article describes how to create user-defined attributes using Rep++ studio, how to associate them to Rep++ objects and how to access UDA values programmatically[1].

[1] *The products and software mentioned in this document are trademarks, registered trademarks or trade names of their respective holders.*

Extending the Rep++ Metadata with User-Defined Attributes

User-defined attributes

The Fields, Rowsets, RowsetTrees, nodes, and other objects are defined in Rep++studio using a number of attributes such as type, control type, min and max size for Fields, options in Rowsets, etc. The attribute values are used to present and validate your data.

A user-defined attribute (UDA) is a metadata extension. It adds a criterion to the definition of the target object, and can be used for presentation or validation purposes, just as standard metadata.

UDAs are included in liststhat specify the sections where the attributes will appear. A list can identify one or several sections, even sections in different systems within your connection, and contain as many attributes as you need. All attributes must be included in such a list.

UDAs are stored in separate tables in the repository, but those tables are not linked with any other exportable component (system, section, program or module). This means that if you move the repository from a development environment to a production environment, then you also need to move explicitly the attributes tables. This is achieved using the import/export functions of the user attributes editor.

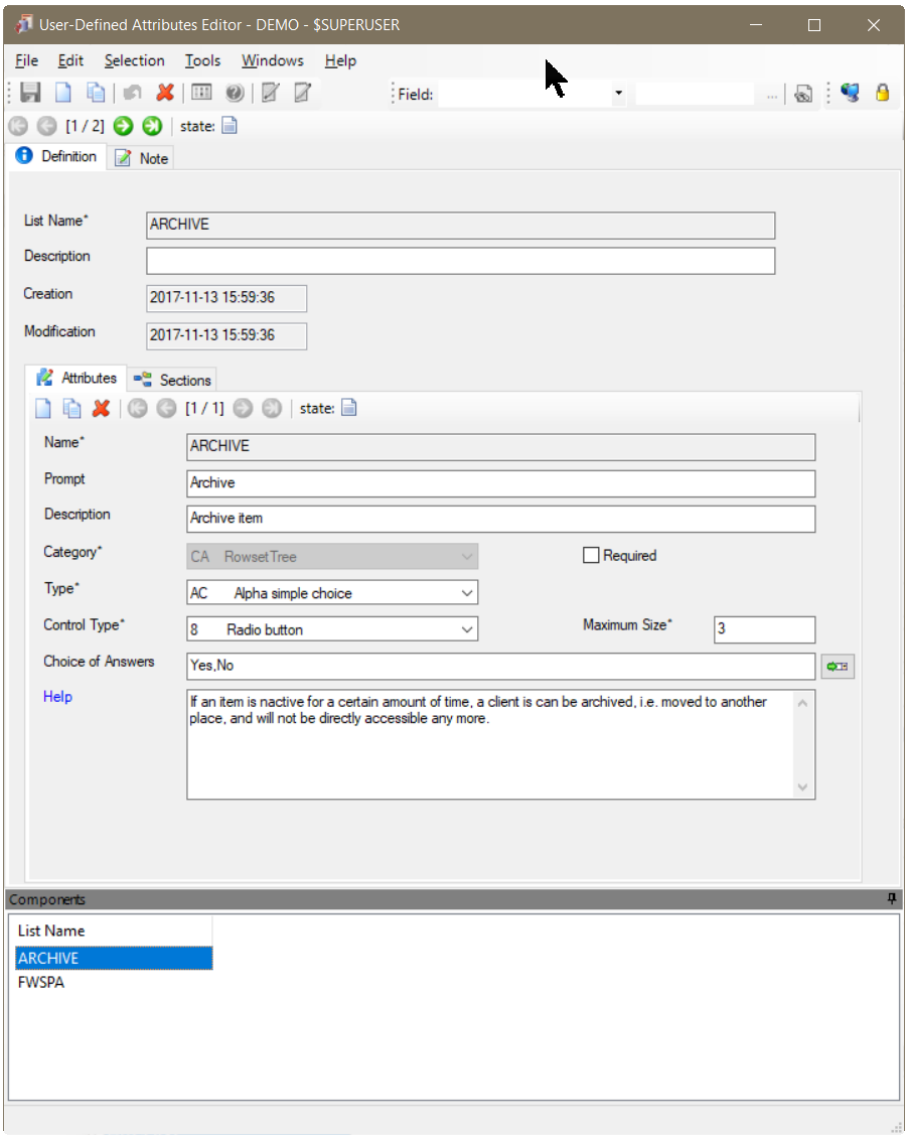


Figure 1. The User Attributes Editor.

Creating user-defined attributes

UDAs are created using the Rep++ User Attributes Editor, which can be opened from the startup toolbar of Rep++studio or from the **Tools** menu of the Rep++ repository editor. When you create a UDA, you enter the following information in the editor.

Table 1. Description of the fields of a UDA.

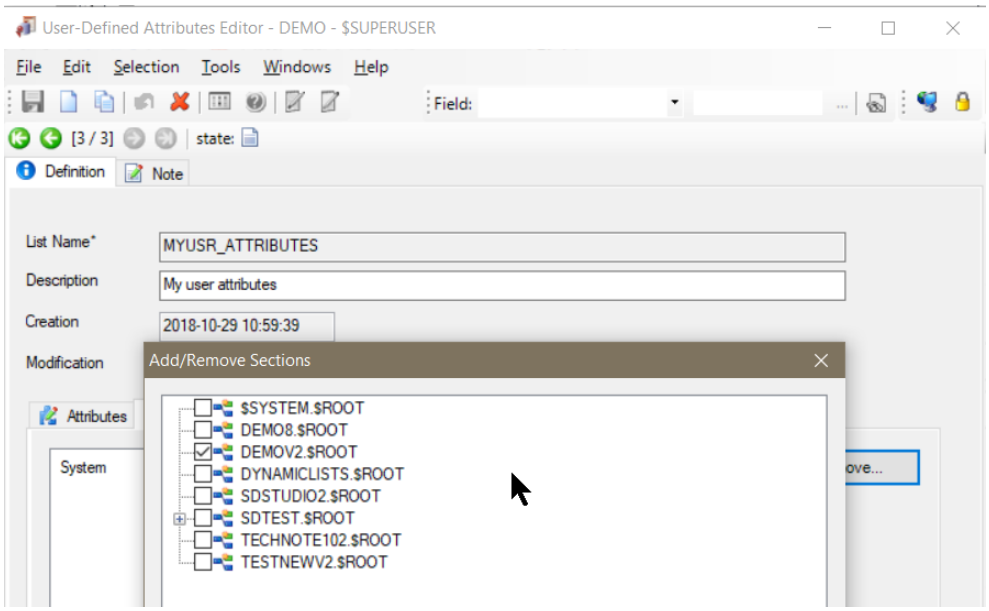
Field	Description
Name	ID of the UDA. Used mainly to access the UDA programmatically.
Prompt	Label of the UDA in Rep++studio.

Field	Description
Description	Description of the UDA (for documentation purposes).
Category	Rep++ object the UDA is associated with (Field, Rowset, RowsetTreeDefNode, RowsetTree, etc.). The Field (Rowset) category is a field that is associated with a Rowset, while the Field (System) category is a field that is not associated with a Rowset, typically used to hold calculation results or information not saved in the database (work field).
Type	Data type of the UDA value.
Maximum Size	Maximum number of characters for the UDA.
Required	Indicates if the UDA value is required. This is enforced by Rep++ when an item with an associated UDA is about to be saved.
Control Type	Type of the control used to display and edit the UDA value.
Choice of Answers	List of choices for the UDA value. Click the icon at the right to help you build the choice list.
Help	Help text to be displayed in Rep++studio when the end user presses F1 while the UDA control is the active control.

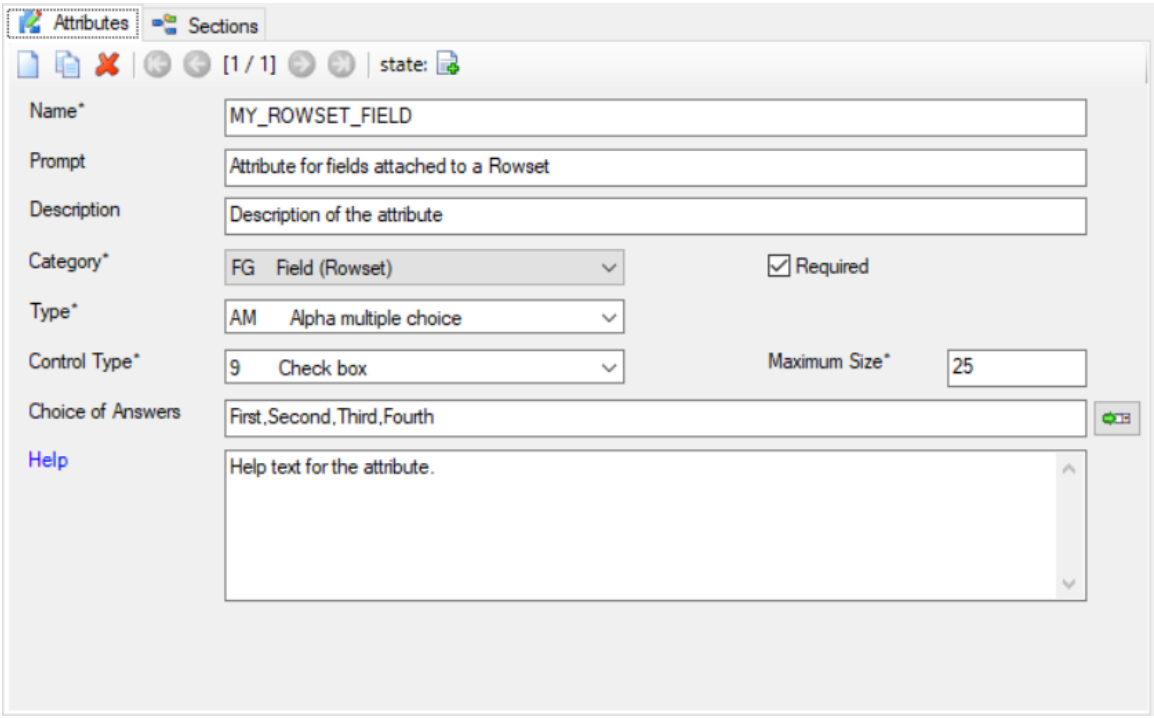
To illustrate the concept, you will create a UDA for a field in a Rowset in the contact management application.

To create a UDA

1. Open the User Attributes Editor using the connection where your contact management is located.
2. Create a new list named MYUSR_ATTRIBUTES.
3. Click the **Sections** tab.
4. Select the DEMOV2.$ROOT section. Click **OK**.

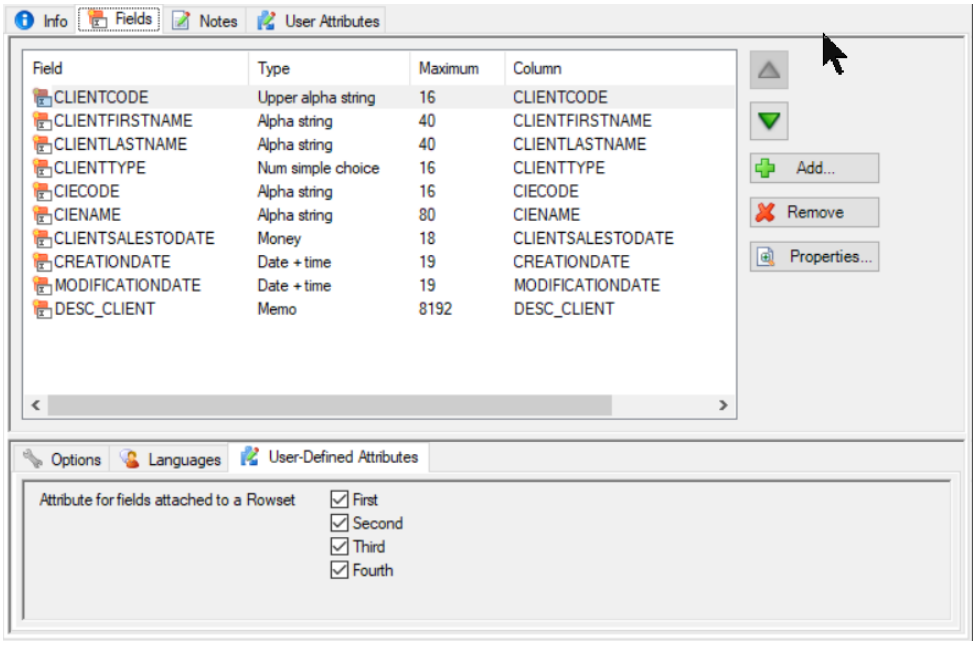


5. Click the **Attributes** tab.
6. Click the **New** button on the **Attributes** toolbar.
7. Enter the information as illustrated in the following figures and save.



8. Restart Rep++studio to make your attributes available.

Your UDA now appears as if it were standard metadata in Rep++studio.



Accessing the values of user-defined attributes programmatically

Once you have created your UDAs and set their values in Rep++studio, you can access them programmatically and get or set the attribute values. First, set the UDA value of the CLIENTCODE field that is attached to the CLIENT Rowset by selecting one or more choice.

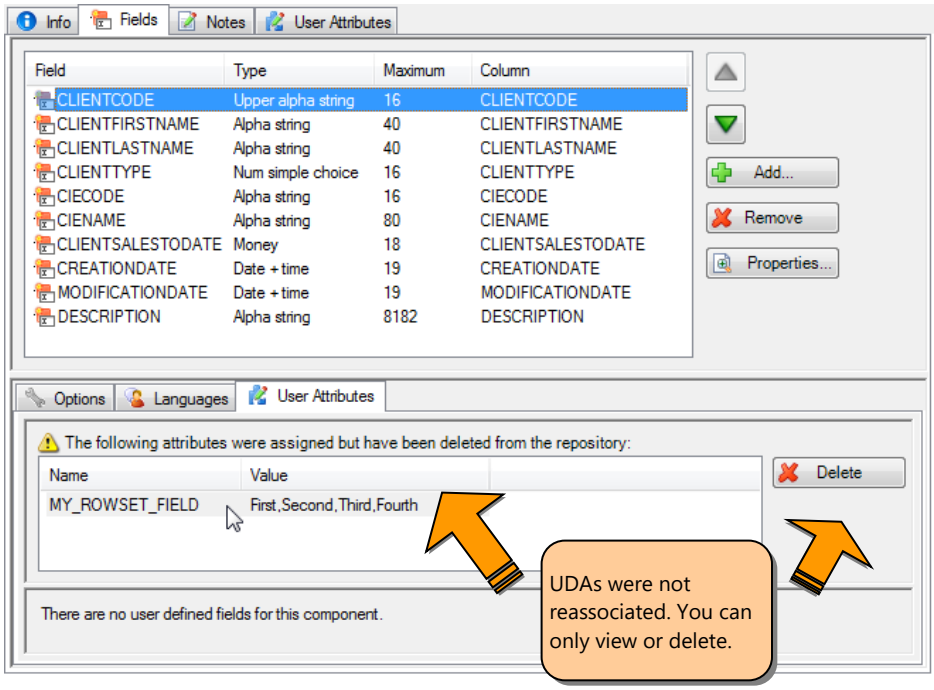
The value of the MY_ROWSET_FIELD UDA associated with the field CLIENT.CLIENTCODE should be "First,Second,Third,Fourth". To get or set this value in your program, use the Attribute.Value property for the MY_ROWSET_FIELD attribute of the CLIENTCODE FieldDef object.

```
using System;
using System.Windows.Forms;

namespace Scrap {
    public partial class Form1 : Form {
        public Form1() {
            RepPP.Application   app;
            RepPP.FieldDef      fld;
            RepPP.Attribute     uda;

            InitializeComponent();
            app     = RepPP.Application.CreateFromRes();
            fld     = app.FieldDef("CLIENT", "CLIENTCODE");
            uda     = fld.Attributes["MY_ROWSET_FIELD"];
            if (uda != null) {
                MessageBox.Show(uda.Value); // First,Second,Third,Fourth
            }
        }
    }
}
```

When you run your program, a message box displays the value of the MY_ROWSET_FIELD UDA associated with the CLIENTCLODE field within the CLIENT Rowset.

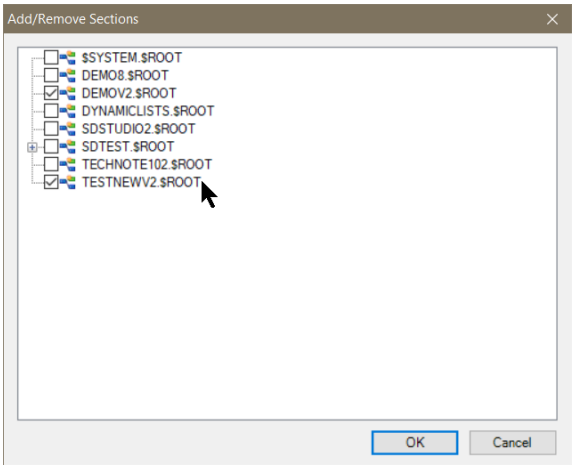
Deploying systems that include user-defined attributes

User-defined attributes are stored independently from the systems in which they are used in the repository. They are not automatically exported nor associated with them. So when you export a system that uses UDAs and then import it into a different repository or in the same repository but under a different name, you need to explicitly re-associate your UDAs. Otherwise, you will only be able to view the current value of UDAs or delete the UDAs associated to the currently selected Rep++ object, with no possibility to edit them. Rep++studio loads the definition of the UDAs associated with a section to create the user interface control used to edit its value. It then loads the value of the UDA in the control. If it cannot access the definition of the UDA in the section, Rep++studio cannot create the control, and therefore no editing is possible.



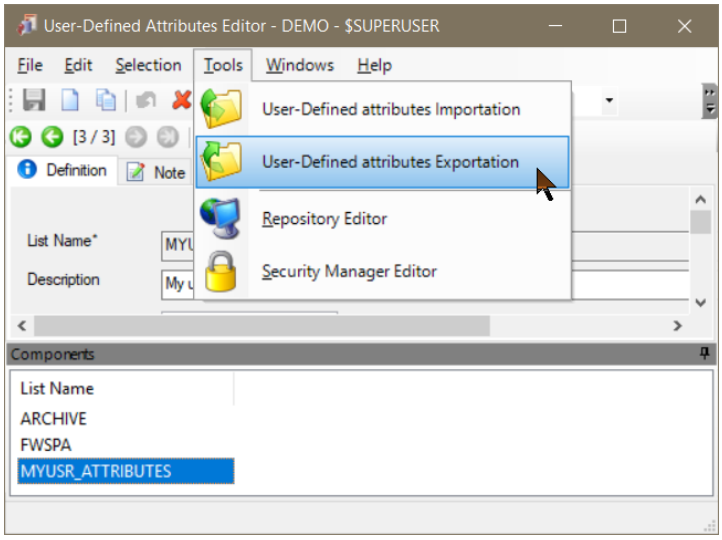
In order to be able to edit the values of the UDAs then make sure you do the following, according to your case:

- **You imported the system under a different name in the same repository**. Since the UDAs are already stored in the repository, all you have to do is to associate the UDA list with the new **Section** name.



- **You imported the system in a different repository**. The UDAs do not exist in the new repository, so they have to be exported from the original repository and imported in the new one.

  a. Export your UDA list from the old repository using the **User attributes Exportation** command.



b. Import your UDA list into the new repository using the **User attributes Importation** command.
c. Associate the UDA list with the sections that need the UDA.

Conclusion

You have seen how you can extend the metadata of the Rep++ repository for Rep++ objects. Most of the time, you will be able to accomplish your tasks simply using the standard metadata that is already available. There are no specific situations where UDAs should be used: it all depends on application requirements and your judgment as an application designer.