Technical Note 101

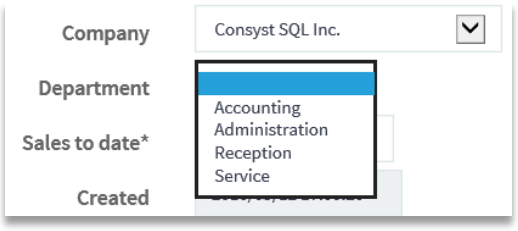
Implementing an auto-refresh choice list

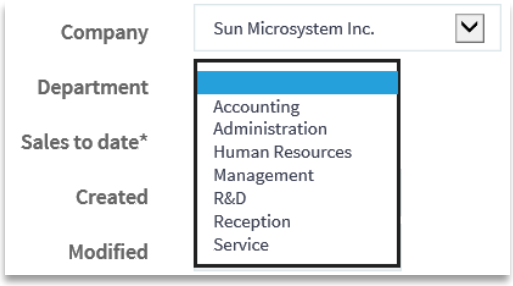
Overview

You may want, in your application, for an end user to choose an item from one list, then have a second list adapt dynamically its content to the value chosen in the first list, without reloading a page. For instance, the list of classes offered depend on the selected faculty or department, and the list of company departments may vary according to the selected company.

Rep++ provides an auto-refresh feature that is easily associated with a choice list in your application. The feature is implemented through a user-defined attribute available for SPA.

This article describes how to incorporate an auto-refresh choice list whose content depends on the value of another choice list.[1]

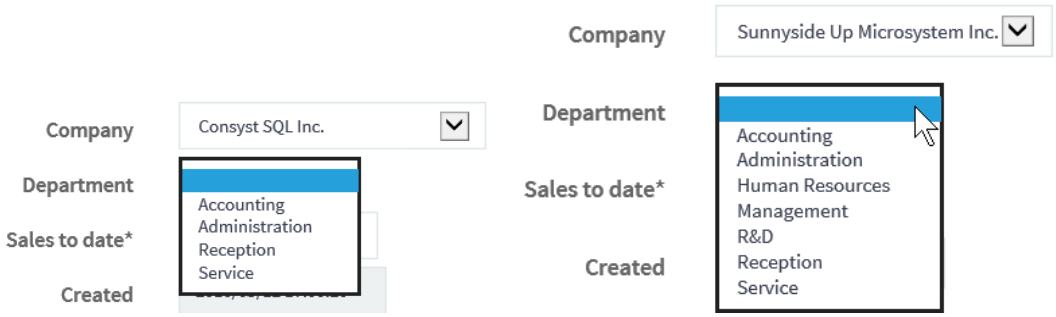




[1] *The products and software cited in this document are registered trademarks, trademarks or trade names of their respective holders.*

Implementing an auto-refresh choice list

To illustrate this feature, you will include two lists in your main Rowset, Companies and Departments. Each list is built using a SQL command, but the list of departments will depend on the value chosen in the Companies list. The auto-refresh is implemented using a specially designed user-defined attribute, but you can also implement it programmatically.



Sample system

A sample system designed to demonstrate this capability, TECHNOTE101, is included with the Rep++ installation. The information is found in 4 tables, TN101_CLIENT, TN101_COMPANY, TN101_ DEPARTMENT and TN101_COMPANYDEPARTMENT. The tables contain the following columns:

Company table

Ciecode	Ciename

Department table

Depcode	Depname

CompanyDepartment table

Ciecode	Depcode

Client table

Clientcode	ClientFirstname	Clientlastname	Clienttype	**Ciecode**	**Depcode**	Clientsalestodate	...

The functionality will be tested in a single-page application (SPA).

In summary, you will:

1. Create the technote's database tables and import the system.
2. Import the user-defined attributes that implement the auto-refresh feature.

3. Set up the Ciecode and Depcode choice lists in Rep++ studio.
4. Create a new SPA application in Visual Studio® using the Rep++ wizard.
5. Test your application.

Prerequisites

Rep++ installed (includes Rep++ studio and SD Tools). Working knowledge of Rep++. SQL Server as database.

Create the Technote101 database tables and import the system

The tables for this system will be added to the repository for the chosen connection.

To create the Technote101 database tables

1. Open SD Tools.
2. Double-click the connection where you want to test the auto-refresh feature. A window with your connection contents opens.
3. On the **File** menu, click **Open**.
4. Choose the Demo/Technotes/Technote101_Srv.sql file under the Rep++ installation folder.
5. On the toolbar, click **Execute**. Your tables have been added.

To import the Technote101 system in your connection

1. In your connection window, in the right pane, double-click **Import a system**.
2. Choose the Demo/Technotes/Technote101.sys system under the Rep++ installation folder and click **Open**.

The system has been added to your connection. You can close SD Tools.

Import the user-defined attributes

You may already have imported the FWSPA user-defined attributes in your connection through a previous exercise. If this is the case, skip the import and include them in the section where you want them, in this case, Technote101.

To import the udas

1. Open the Rep++ User-Defined Attributes Editor.
2. On the **Tools** menu, click **User-Defined Attributes Import**.
3. In the *User-Defined Attributes Import* dialog box, select the SPAUDADEF.uda file located under the Rep++ installation folder, in the Rep/base subfolder. Click **OK**.

The set of user-defined attributes that implement the auto-refresh functionality are included in the FWSPA component, which now appears in the components list.

In order to use the auto-refresh feature, you must explicitly specify the Rep++ section(s) where the FWSPA user-defined attributes will be available.

To include the user-defined attributes in a section

1. In the *User-Defined Attributes* editor, select the FWSPA component and click the **Sections** tab below the definition section.
2. Click **Add/Remove**.
3. Check the section containing the program where you want to use the auto-refresh and click **OK**.
4. Save your modifications.
5. Restart Rep++ studio for the changes to take effect.

Set up the CIECODE and the DEPCODE choice lists

The TECHNOTE101 system contains a Client Rowset in which the CIECODE and DEPCODE are included, both of which are dynamically built lists. The content of the DEPCODE list depends on the CIECODE list.

You first need to define the SQL commands to build the CIECODE list and the DEPCODE list, then modify the settings of the fields within the Rowset, activate the auto-refresh function, and finally test your program.

To create the CIECODE and DEPCODE choice lists

1. In the SQL commands editor, add an SQL command atom named List_Companies containing the following code:

```
SELECT TN101_COMPANY.CIECODE, TN101_COMPANY.CIENAME
INTO :$CODE, :$DESC
FROM TN101_COMPANY
ORDER BY TN101_COMPANY.CIENAME
```

2. Add another SQL command atom named List_Departments containing the code that will link its content according the the selected company:

```
SELECT TN101_DEPARTMENT.DEPCODE, TN101_DEPARTMENT.DEPNAME
INTO :$CODE, :$DESC
FROM TN101_DEPARTMENT
INNER JOIN TN101_COMPANYDEPARTMENT ON DEPCODEFK = DEPCODE
WHERE CIECODEFK= :$AUTO.CIECODE
ORDER BY TN101_DEPARTMENT.DEPNAME
```

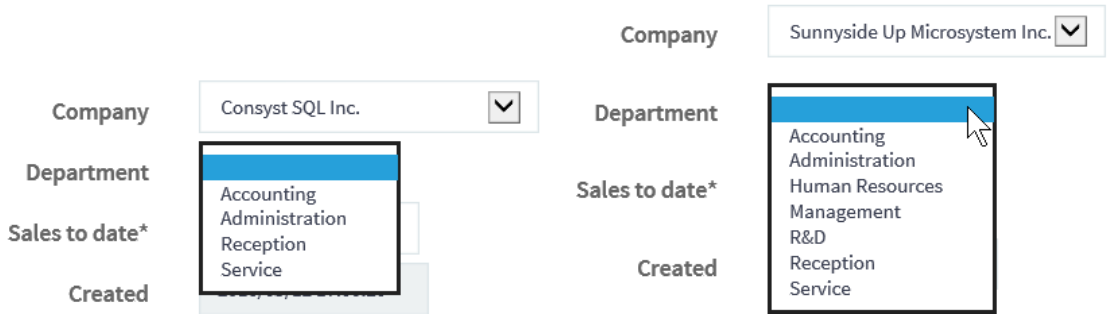
3. Click the **Fields** node in the **Repository Components** tree.
4. Open the CIECODE field and change the following attributes:
   - Control type: Combo box
   - SQL command for list: List_Companies
5. Save your modifications.
6. Open the DEPCODE field and change the following attributes:
   - Control type: Combo box
   - SQL command for list: List_Departments
7. Save your modifications.

To modify the Rowset settings and activate the auto-refresh feature

1. Click the **Rowsets** node in the **Repository Components** tree.
2. Open the Client Rowset.
3. Select the CIECODE field from the list.
4. In the **Options** page, under **Join/Choice list** options, check **Choice list by SQL cmd**.
5. Select the DEPCODE field from the list.
6. Click the **User-Defined Attributes** tab below the fields list:
   - Click **Yes** next to the **Dynamic choice list** attribute. (You can do this step programmatically, as explained below.)
7. Save your Rowset.

You can now create a Rep++ SPA application to test the auto-refresh functionality of the Department field (see **Create a Rep++ SPA application to test your functionality** at the end).

Notice the variation in the department listing for each company.



Programmatically activate the auto-refresh function

Alternately, you can replace steps 6 above by setting the IsAutoRefresh property of the DEPCODE field.

To programmatically activate the auto-refresh function

1. In the Visual Studio® Solution Explorer, expand the nodes ProjectName\Scripts\SpaModels\ClientTransaction and open the RTCClientTransaction.ts file.
2. Locate the ccAfterLoad method and add the line in bold:

```
public ccAfterLoad(): void {
    super.ccAfterLoad();
    this.rsClient.rsDefClient.fldDefDEPCODE.isChoiceListAutoRefresh = true;
}
```

3. Save you modification.
4. Rebuild and run.

Create a Rep++ SPA application to test your functionality

To test your new functionality, create a Rep++ SPA application using the Rep++ wizards.

To create a Rep++ SPA application using the Rep++ wizard

1. In Visual Studio®, create a standard ASP.NET Web Application (not Rep++).
2. In the *New ASP.NET Project* window, select the **Rep++ MVC** template.
3. In the Rep++ wizard's *Connect to the Rep++ repository* page, select the connection, system and program for your technote.
4. In the Select *Rep++ repository components* page, select the *ClientTransaction* and *ClientSelectTransaction* RowsetTrees for the transaction and selection buffer, respectively.
5. In the *Select n-tier options* page, leave the default values.
6. In the *Specify generation information for typed instances* page, leave the default values.
7. In the *Specify generation information for POCO entities* page, clear **Generate POCO entities**.
8. In the *Select template* page, select *MVC SPA using Angle Template*.
9. In the *Select layout options* leave the default values.
10. In the *Select replacement mode* page, click **Finish** to create the application.
11. In the *Search for TypeScript Typings* message box, click **No**.