**CC Table Tree**

Documentation

22. Dezember 2014

Christian Coppes

Table of Contents

[1 About CCTableTree 3](#_Toc407047302)

[2 Different Versions 5](#_Toc407047303)

[2.1 DAO Internal 5](#_Toc407047304)

[2.2 DAO External 5](#_Toc407047305)

[2.3 ADO/ADP External 6](#_Toc407047306)

[3 Table Tree Form 7](#_Toc407047307)

[3.1 Details about the form 7](#_Toc407047308)

[3.2 Events 7](#_Toc407047309)

[3.3 Multi Tests 7](#_Toc407047310)

[4 Configuration Editor 8](#_Toc407047311)

[4.1 Default configuration 8](#_Toc407047312)

[4.2 Configuration for all levels 8](#_Toc407047313)

[4.3 Different configuration for each level 8](#_Toc407047314)

[4.4 Editing the RTF fields 8](#_Toc407047315)

[4.5 The configuration columns 9](#_Toc407047316)

[4.5.1 Configuration Name 9](#_Toc407047317)

[4.5.2 User ID 9](#_Toc407047318)

[4.5.3 Assign Cfg. To Level 9](#_Toc407047319)

[4.5.4 Handle Opened/Closed 9](#_Toc407047320)

[4.5.5 Textfield Format 9](#_Toc407047321)

[4.5.6 Vertical Line / Branch Line / Corner Formatted 9](#_Toc407047322)

[4.5.7 Level Format 9](#_Toc407047323)

# About CCTableTree

CCTableTree is a system of some modules, tables and forms which allows you to create a tree view of a table which is organized in a hierarchy way, where you have a numeric (Long Integer) ID as row primary key and another column (also Long Integer) which points to the parent ID.

Normally you would use the Microsoft Common Controls, the TreeView ActiveX object, to render such hierarchy. Although that is a great control it has several disadvantages:

* You have only one node text to display
* You need to extract the data from the table and create a tree in the way the ActiveX component understands it which can take a lot of time if you’re new to this
* The MS Common Control library is not guaranteed to be installed on the target computer – or not in the right version
* In 2012 a defect update from Microsoft of this library made all programs which used one of this controls unusable
* It’s not sure how long these controls will exist in the future

As I needed to have a tree in a table form which works with standard Access continous forms and the need to avoid the MS Common Controls library in the current project I developed this system which is able to work in any kind of Access database, as single database file, as frontend/backend solution or in an ADP (Access Database Project in conjunction with a SQL Server, possible up to Access 2010).

This system has the following features:

* A row column from 1 to x for all nodes of a tree
* A selection column where each node can be selected, including the possibility to select all subnodes or deselect the parent node automatically (in the way the old Microsoft installers displayed the feature selection)
* An RTF formattable handle icon (i.e. a symbol character from a symbol font, but don’t need to be a symbol) which shows the state of the current node if it has children and if they are expanded or not, this can be clicked to open or close the children nodes
* A method to expand or collapse all children beginning with the current one
* An RTF formattable level string which shows the level of the current node, additionally the option to add an ASCII Unicode value to the level number to show special font characters like ① , ②, ③
* The possibility to show or surpress the numeric level numbers with the “Show Level No.” option if you want to display a fix level text only (or even nothing)
* Separated RTF formattable line characters to build a tree like this (also included in this example another way of displaying levels with the ASCII Unicode value addition):  
  -Ⓐ-   ┣ Test Node Entry  
  -Ⓑ-   ┃ ┣ Test Node Entry 1  
  -Ⓑ-   ┃ ┗ Test Node Entry 2  
  -Ⓒ-   ┃ ┣ ┗ Test Node Entry 3  
  -Ⓐ-   ┗ Test Node Entry 4
* An RTF formattable node text entry where you can also add fix prefixes and suffixes if you want
* The possibility to JOIN any table to the tree table so you can display **and** change the values of the JOINed columns in the original table (must be an INNER JOIN where for each node there is only one record in the JOINed table)
* An additional Custom ID field which is saved only in the tree table (not the original table) for your own purpose
* The option to select a maximum level which should be loaded into the tree
* Quick selection of a tree style using a combobox
* One default setting (applied to all levels) and an unlimited number of additional settings, either as general setting or saved together with the User ID as individual setting of one user only
* Either one configuration line for all levels or more than one if each level should be displayed in a different way
* Automatic assignment of the “0”-level configuration line in a multiline-configuration if more levels exists in the tree than configuration levels exists
* Events which are fired if the user changes a selection checkbox or if another node row is clicked which delivers the node ID to the event consumer, events “BeforeNodeOpen/Close” with a Cancel parameter which makes it possible to forbid opening or closing a node and “OnNodeOpen/Close” when the node is opened or closed
* The tree can be reloaded if the tree GUID is specified to the Init procedure so the tree can be displayed faster without complete rebuild
* Saveable and loadable configurations in a configuration table
* Trees are maintained separated by a Long Integer User ID and an additional Tree GUID so the same user can use several tree forms at the same time and also several users can use the same tree table (if you have a central tree table database file or a central frontend file) – but it’s also possible to give any user an own table tree for his local frontend
* A comfortable configuration editor to enter all these things into a configuration table including an RTF preview field and a text editor for the RTF fields which allows to change the tags manually (i.e. to enter hex color numbers which the Access built-in mini RTF editor doesn’t offer).
* The tree form only exists to demonstrate the features, the tree classes are flexible because you can define the control names of the form to be used on your own and also the layout so you can display your tree in the way you want (the tree can also be displayed in the Datasheet View with full formatting, but without the possibility to open/close the subnodes by clicking on the handle)
* The tree uses my “Reposition” class code so the tree is requeried without changing the record and screen position
* The system is free to be used in any database project you want as long as you do not try to sell these as your own invention. You can also change the code to fit your needs as long as the informations about the author(s) kept unchanged.

# Different Versions

There are three different versions to demonstrate the different kind of usage of the CCTableTree system.

## DAO Internal

The DAO Internal version shows the usage of the system if you want to have all in one frontend file (or single database file). They contain the 1.0 class versions of the system which can only be used in this kind of database usage. Here the tables “tblCCTableTree”, the “tblCCTableTreeConfig”, the form “frmCCTableTreeConfig” and the classes “clsCCTableTree” and “clsCCTableTreeConfig” must be inserted into the own database where you want to use them. The other objects from the demo database are the data tables to show the usage with two different tree tables, a test module which shows how to open the “frmDemoTableTree” with more than one user and/or more than one tree and the “frmDemoTableTree” which shows how to display a tree and how you can consume the events the class fires.

Moreover there is a little module “modVBA” which I used to create table and field constants and properties. You can use that in your own projects, if you like. But that has nothing to do with the CCTableTree system, you don’t need to import that.

## DAO External

The DAO External version uses the 2.0 class versions of the system. The difference is that these classes searches for an external database file named “CCTableTreeData.mdb” which contains the tables “tblCCTableTree”, “tblCCTableTreeConfig” and the form “frmCCTableTreeConfig” which is the configuration editor.

The demo form “frmDemoTableTree” and the module “modCCTableTree” remains in the normal frontend and of course the two classes. The “modCCTableTree” contains replacements for DCount and DMax which are used in the SQL of the table tree classes.

This is needed because the domain aggregate functions (DCount/DMax and so on) are not part of the JET/ACE database engine but part of the Access application. If you open a database file without linking it to the current database the aggregate functions doesn’t work even if you use the right workspace object because they are called locally and searches for the tables in the local workspace only. So these replacements (“XCount” and “XMax”) in the module are needed to perform a standard SQL method on the external database file.

Why is it needed to use aggregate functions? This is because Access unfortunately is not able to use Sub-SELECTs inside an UPDATE query. If you use them, the SELECT is not updatable anymore. As a replacement you must use a VBA function or the domain functions to get the data you need, then the SELECT remains updatable.

When the tree is built, the tree table is filled in the external database file and so the frontend will not grow as it would in the “DAO Internal” version. The table tree is heavily used and so often loaded with data and deleted, that means, without compacting the file will grow bigger. It is set to automatically compact on close but you need to open it by yourself and close it, simply opening a connection and closing it doesn’t compact the database file.

Another difference to the V1 class is that the V2 class has a conditional compiling constant at the top of the class code. You can set that to “ADO” or “DAO”. In the “DAO External” version it is set to “DAO” to compile anything with a DAO frontend. See below in “ADO/ADP External” when you need to switch that to “ADO” instead.

## ADO/ADP External

The ADP file in this folder has no server connection. To use it you must connect that to any SQL Server which contains the usual “AdventureWorks” database which contains the “HumanResources.Employee” table used in this demo. The other little tree table which is in the frontend in the other versions can be created and filled with data if you use the “CreateTblTree1.sql” script in this folder.

The V2 class also available in this class has the conditional compiler constant “ADOorDAO” set to “ADO” so this can be used with an ADP. That is used to fill the tree table using ADO, nevertheless, the external tree table file is used as a DAO database in the same way as in “DAO Internal” above. You often hear that an ADP file cannot use local tables. That is of course true for the ADP file itself but this sample shows also that you can of course use local tables if you use a second file. It also shows that you can JOIN between DAO and the SQL Server database tables which is also updatable – using an ADP to show the DAO recordset. That’s possible because Access forms can use both kind of recordsets either in ACCDB/MDB and also in ADPs.

Except that the class now works in an ADP and that the conditional compiler constant is “ADO” here there is no difference between DAO External and ADO External, it works the same way.

# Table Tree Form

## Details about the form

The demo table tree form should only show how to use the CCTableTree system. You do not need to import that to your own database if you don’t want, you can completely build your own form because the class has an Init procedure which you can provide with any reference and names needed to use the class with other forms, too.

To use the click function on the handle it is needed to design a transparent button over the handle textbox. On this way it is possible to click on the handle without marking the complete textbox and without a cursor appearing in the textbox. It needs to be a textbox because only that control is able to show an RTF formatted contents, the same is true for the TreeNodeText and Level fields, otherwise you would see the HTML tags in these fields.

You can of course also use a normal designed button beneath the handle and simply set the handle textbox to locked and deactivated, but that doesn’t look so impressive as directly clicking on the handle. You need that method if you want to use an older version than A2007 because transparent buttons were introduced in A2007.

The controls in the header shows the other methods possible with the class, like limiting the maximum number of displayed levels, selecting the style and switching the tree to another one. The expand and collapse buttons starts at the current selected node in the tree.

## Events

The form shows how to use the events the class fires. Don’t wonder that you cannot open the branch "adventure-works\stephen0" of the Human Resources tree. The “Cancel” parameter can be used to block opening the current branch. In the same way the “BeforeNodeClose” event with it’s “Cancel” parameter shows that you can also open a node but not close it again. That’s the case with “adventure-works\amy0" node.

## Multi Tests

The “modTest” module demonstrates how to use the same form to open either two trees for the same user or the same tree by two users using the same external or internal tree table. Before starting any if these procedures you should comment the SelectTree call in Form Load of the demo form, otherwise this would create the tree once more for each form.

# Configuration Editor

## Default configuration

The default configuration “\_Default” must not be deleted from the table, this is used if no other configuration is assigned to the current tree. Also the “UserID 0” and “Assign to Level 0” must not be changed. The other fields are open to be changed, so you can configure your own default configuration if you want.

## Configuration for all levels

If you want to create a configuration which will be assigned to all levels of the tree then set the “Assign Cfg. to level” value to 0. That’s all for this variant.

## Different configuration for each level

This variant is useful if you want to design a tree where every level is displayed in a different way. You are free to design each level completely independent of the level before. To achieve that, each level number must be entered in the “Assign Cfg. to level” column, so 1 for level 1, 2 for level 2 and so on, one row in the config table for each level. Create an additional “0” row for any further level, i.e. if your tree has 6 levels but your config only goes up to level 3.

The configuration is grouped by the class using the identical configuration name and user ID, where “0” is for general configurations and any other for a specific user ID from your user maintenance.

## Editing the RTF fields

All RTF fields using a kind of “light version” of HTML. Unfortunately only some few tags of HTML are usable in Access RTF fields. You can directly edit the RTF fields in the list using the Access mini RTF editor which appears when you edit a field. The disadvantage is that you have no control over what it saves into the field. Moreover the number of colors is limited to the number of colors displayed in the mini editor.

For this reason there is a text version of the field displayed in the footer of the form, whenever you enter an RTF field. You can also edit the contents here, it will change in the field above at the same time.

You’ll see that the field is always surrounded by “div” tags, which Access automatically inserts. Don’t worry, these are automatically stripped off the field when the tree is shown (otherwise the div tags would create a new paragraph in the output).

The advantage of entering the field as text is that you have full control over the contents, you can remove unneeded texts and you can enter any color you want using the hexadecimal version like “#FFFFFF” for white. Don’t use color names like “Red” or anything like this although the RTF field is able to display that. The tree classes needs this hexadecimal style to replace vertical lines with the background color (that’s the trick to display the tree lines with spaces between some nodes, you see that if you mark such a node). In other RTF fields except the “Vertical Line Formatted” field it doesn’t matter if you use hex colors or named colors.

Moreover there is a little preview field showing the result of your configuration for one level (for multiple levels you must use the demo form to see the result).

## The configuration columns

### Configuration Name

Any configuration need to have a unique name in the configuration table. If you have a multi-level config (one config for each level) you must repeat the same name in each config line with a different “Assign Cfg. To Level” value. The config name is unique inside of one user ID, so different users can create the same configuration name for them.

### User ID

The user ID is a long integer which should come from your user maintenance (a login table etc.). This can be used to separate lists of configurations for different users. If the user ID is 0 then the configuration can be used by all users. There is a “FillConfigCombobox” method in the “clsCCTableTree” which allows you to automatically fill a combobox either with the general or additionally with the config of a specified user ID.

### Assign Cfg. To Level

Use the value 0 for a default configuration in a multi-level config or always in a single-level config.

Use any higher number for the desired level. This makes it possible to have different ways of displaying a tree, i.e. you can have different colors or font sizes for each levels, but also it is possible to have a config for only one specific level and all others using the 0 config.

To group the configurations together the same config name must be entered with the same user ID and different level numbers.

### Handle Opened/Closed

These two fields are used to display a different handle for all nodes which have children nodes. Depending of the state of such nodes either the “Opened” or “Closed” handle string is displayed. Feel free to experiment with any font or style you want. It can also be one or more characters (the demo form would display only one because it is so small but you can make the handle field larger and display longer handle strings).

### Textfield Format

The textfield format must contain the string “{#}” which will be replaced by the node text later. The normal way would be to only insert this string here and format it in any style you want, but you can of course add a fix prefix or suffic before and after this string which would be displayed on any node.

### Vertical Line / Branch Line / Corner Formatted

These three fields are used to build the tree lines. Normally this consist of one character only, but it is not limited. In case of the vertical line you need to make sure that the text is surrounded by only one font tag which must contain a color in hexadecimal format like “#FFFFFF” because the class searches for this color whenever the line should be hidden and replaces it with the textbox background color (which cannot be transparent for correct display).

You should make sure that also the other line fields doesn’t contain more than one “color” attribute.

### Level Format

The level field must also contain the text “{#}” which will be replaced by the level number. It can, like the textfield format, also contain a fix prefix or suffix text, also with different colors and so on.

The field will display the level number if the field “Show Level No.” is checked also. If not, the “{#}” is replaced by an empty string. That can be used to display a fix text instead of a level number like shown in the “Human Resources” style.

Additionally there is a field “Add ASCII To Level” which can be used to add a unicode ASCII value to the level number. So “9311” as in the demo would be result in “9312” for level 1, “9313” for level 2 and so on. That parameter is useful to use special characters in a font like 9312 which is a circled 1, where the other characters following are a circled 2, 3 and so on. You can also use this if you design an own font with special characters to be displayed for each level number, the only thing which is needed is that they must be designed beneath each other (like with 9312, 9313, 9314 and so on). The circled numbers in the used font here are limited up to level 9, but in the “User 3” style you can see that also circled A up to circled Z can be used to display levels which gives a greater range of possible levels.