Hierarchical Lists in 4D v11 SQL

By Thomas Fitch, Technical Support Engineer, 4D Inc.

Technical Note 07-39

Abstract

Hierarchical Lists have been dramatically improved in 4D v11 SQL. This Technical Note explores the changes made and demonstrates how to use the new features of Hierarchical Lists.

In this document there are references to different pages of the sample database. To navigate in the sample database between pages use the "Goto Previous Page" and "Goto Next Page" buttons on the display form.

Updated Commands and Implementations

This section discusses commands that have been updated for 4D v11 SQL and explanations and examples of the updates. It also includes general implementations to the Hierarchical List topic as a whole.

Multiple Form Objects and Commands Affected

In earlier versions of 4D Hierarchical Lists each had a list reference ID. This allowed developers to reference the list as a language object and had object names which allowed the list to be referenced as a form object. The problem with this was that each language object could only be displayed as one form object, so if a developer wanted to display the same Hierarchical List multiple times (on the same or different forms) they had to rebuild the list each time and store an instance of it in memory.

With 4D v11 SQL both of those methods of referencing a Hierarchical List remain, but now the same language object can be tied to multiple different form objects. This allows one list to be built and displayed in different positions and on different forms via separate form objects. Many of the characteristics of these separate form objects stay the same from list to list, but a few of them can be unique for each separate list:

- The currently selected items of the list.
- The expanded and collapsed states of items in the list.
- The position of the scrolling cursor in the list.

This new function of the form objects of Hierarchical Lists calls for an update to the syntax of many commands. When a developer wants to reference a specific form object, rather than all form objects related to a list ID, there needs to be a way to pass that object as a parameter. To do this a new optional parameter has been added to many commands.

Commands which used to look like this:

COMMAND (*listrefID*; *parameter*; *parameter*)

Can now be written this way instead:

COMMAND (*; *"objectname"*; *parameter*; *parameter*)

Commands which use this new parameter include:

SET LIST ITEM
GET LIST ITEM
DELETE FROM LIST (changed from **DELETE LIST ITEM**)
SELECT LIST ITEMS BY POSITION
Count list items
Selected list items
INSERT IN LIST (changed from **INSERT LIST ITEM**)
List item parent
List item position
SET LIST ITEM PROPERTIES
GET LIST ITEM PROPERTIES
SET LIST ITEM FONT (new command for 4D v11 SQL)
Get list item font (new command for 4D v11 SQL)
Find in list (new command for 4D v11 SQL)
SET LIST ITEM ICON (new command for 4D v11 SQL)
GET LIST ITEM ICON (new command for 4D v11 SQL)
SET LIST ITEM PARAMETER (new command for 4D v11 SQL)
GET LIST ITEM PARAMETER (new command for 4D v11 SQL)

Note *The commands **DELETE LIST ITEM** and **INSERT LIST ITEM** have been changed to **DELETE FROM LIST** and **INSERT IN LIST**. When upgrading a database to 4D v11 SQL this change will be made automatically during the upgrade process.*

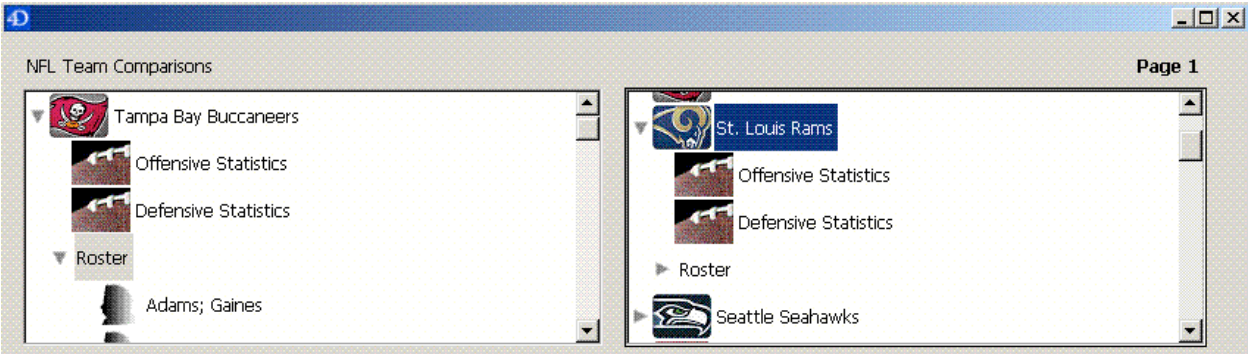
The previous means of referencing the language object via the list reference ID is still accepted and will commonly be used when not referring to a specific form object, but rather the list in general.

Using one of these commands with the new reference parameter may seem to imply that only that form object's properties are changing, but it is important to recall that only the previously mentioned properties are unique per form object. Other properties that are changed for one form object of a Hierarchical List are changed for all form objects tied to the same language object. For example calling **SET LIST ITEM FONT** for one form object could change all representations of that Hierarchical List.

This new means of referencing object by their object name on forms also allows Hierarchical List objects to be referenced with a wildcard character. For example calling **SELECT LIST ITEMS BY POSITION** and using the object name as

reference one could use "HL@" and set a new selected list item for all form objects starting with HL. When using the wildcard character with commands that get values (such as **GET LIST ITEM** or **Find in list**) the first form object whose name corresponds is used.

The sample database is full of uses of these new commands. The demo has two form objects of the same Hierarchical List that can have different selections and different expanded and collapsed list items throughout. Here is one such example:



Note *Using some of these new commands (such as **SET LIST ITEM FONT**) can override properties set in the Property List in design mode or that was set using a command from the Object Properties theme. Using a command from the Hierarchical List theme takes precedence over all other means of setting these properties. Once one of these commands has been used to set a property all other means of setting the property fail. The Hierarchical List commands have the highest precedence and cannot be overridden.*

Commands FONT, FONT STYLE, and FONT SIZE

These general 4D object commands now also work on Hierarchical List objects. They can be called in both ways outlined in the previous section. Using the optional form object name reference a specific representation of a Hierarchical List object can be made to appear differently from other representations of the same Hierarchical List.

This is an effect that cannot be achieved with the Hierarchical List commands and is thus very useful. It is important to remember though that, as noted above, Hierarchical List commands have precedence. A form object that has been adjusted using one of the Hierarchical List commands (such as **SET LIST ITEM FONT**) would not be affected when changing the same aspect of its display with object commands.

There are examples of these commands on page 2 of the demo in the sample database to change font properties via buttons.

Command SET SCROLLBAR VISIBLE

This command can now be used to set the visibility of the scrollbar for Hierarchical List form objects. It can be called using the list reference ID or the form object name as previously discussed.

To test this command in the sample database go to page 1 of the demo and use the "Scrollbars on" and "Scrollbars off" buttons to show examples.

Command SCROLL LINES

This command can now be used to scroll and change the current selection of a Hierarchical List form object. This command can be used with the list reference ID or the form object name, but when using multiple displays of the same list it is usually a good idea to use the form object name and only reference one of the displays. This is because using the position parameter of this command to scroll to a line number only accounts for expanded list items. This means that if one of the lists being referenced has different list items expanded, passing the same value into SCROLL LINES can select a different line for each list.

To test this command in the sample database go to page 1 of the demo and use the different selection and scrolling buttons to show examples.

Command REDRAW LIST

Hierarchical Lists are now redrawn automatically when necessary. The REDRAW LIST command no longer has any effect. When upgrading an older database to 4D v11 SQL this command will not be removed, but it does nothing when called now.

New Commands

In this section new commands that have been added to the Hierarchical List theme will be discussed with examples and explanations.

Commands SET LIST ITEM FONT and Get list item font

SET LIST ITEM FONT ({*; }*list*; *itemRef* | *; *font*)

Get list item font ({*; }*list*; *ItemRef* | *) → *String*

Parameter	Type	Description
*	*	If omitted (default): use list reference ID for list parameter If specified: use list object name for list parameter
list	ListRef (Longint) | String	Name of the list form object if * specified List reference ID number if * omitted
itemRef or *	Longint | *	Item reference number or 0 for the last item added or * for the current item of the list
font	String | Num	Font name or number, return value for **Get list item font** is always font name

These commands get and set the font of a single item in the Hierarchical List. This command does not affect only a single representation of the list, but rather all representations (form objects) of a language object. For this reason it is important to be careful when using the * parameter to reference which list item to set or get the font of. This is because using that parameter means that the developer also needs to use the object name to reference the list if there are multiple representations of the same list. Otherwise 4D will not know which representation's selection of items to use for this command.

It is also good to note that this is an example of one of the commands that overrides object property commands and the Property List in Design Mode. Any changes made with the **SET LIST ITEM FONT** command cannot be changed via those other methods.

With the **SET LIST ITEM FONT** command passing an empty string as the *font* name resets the font to default for the Hierarchical List.

Examples of this command can be seen on page 2 of the demo in the sample database.

Command Find in list

Find in list ({*; }*list*; *value*; *scope*{; *itemsArray*}{; *}) → *Longint*

Parameter	Type	Description
*	*	If omitted (default): use list reference ID for list parameter If specified: use list object name for list parameter
list	ListRef (Longint) | String	Name of the list form object if * specified List reference ID number if * omitted
value	String	Value to be searched for
scope	Integer	If 0: search the main list If 1: search the sublist
itemsArray	Longint Array	If 2nd * omitted: array of positions of items found If 2nd * specified: array of reference numbers of items found
*	*	If omitted (default): use position of items If specified: use reference numbers of items
Function result	Longint	If 2nd * omitted (default): position of item found If 2nd * specified: reference number of item found

This command returns the position or reference number of the first item found to match the value passed into the method and can also fill an array with the corresponding numbers of all matching list items.

The search done is exact, such that the String passed as the *value* to search for must exactly match a value in the list. For example, from the sample database, searching for "Oak" would not find "Oakland Raiders". However this search does support the wildcard character, so in code a developer could add the @ symbol to the end of any searches to change this behavior.

All searches done include a search of the main list passed, but by passing 1 as the *scope* value sublists can be included in the search.

The **Find in list** command is one of the most likely to be used on multiple representations of the same list. An example of this is given in the sample database on page 3 of the demo. Also, code for the **Find in list** command can be found in the sample database in the object methods of the two text variables on that page.

Commands SET LIST ITEM ICON and GET LIST ITEM ICON

SET LIST ITEM ICON ({*; }*list*; *itemRef* | *; *icon*)

GET LIST ITEM ICON ({*;}*list*; *itemRef* | *; *icon*)

Parameter	Type	Description
*	*	If omitted (default): use list reference ID for list parameter If specified: use list object name for list parameter
list	ListRef (Longint) | String	Name of the list form object if * specified List reference ID number if * omitted
itemRef or *	Longint | *	Item reference number or 0 for the last item added or * for the current item of the list
icon	Picture	Icon to be associated with the item (for **SET LIST ITEM ICON**) or icon currently associated with the item (for **GET LIST ITEM ICON**)

These commands get and set the icon of a single item in the Hierarchical List. They expand on the uses of **SET LIST ITEM PROPERTIES** which can also set an icon for list items by allowing the developer to pass non-static picture references to the command use them for their list. The *icon* parameter can be a valid 4D picture expression (field, variable, pointer, etc) but it is suggested to use pointers. This way building a list that uses the same picture over and over would not use as much memory.

The **LIST ITEM ICON** commands are more commands for which it is important to be careful when passing a 0 as the *itemRef* parameter. This is explained above under **SET LIST ITEM FONT** and **Get list item font** commands.

Sample code for the **SET LIST ITEM ICON** commands can be found in the sample database on the form method of the "HL_display" project form where the Hierarchical Lists are created.

Here is an example taken from that method:

```
READ PICTURE FILE (Get 4D folder (Extras Folder)+[Teams]Abbr+".gif";$icon)
APPEND TO LIST (<>HL_level1;$teamname;$id;$sublist;False)
SET LIST ITEM ICON (<>HL_level1;0;$icon)
```

Commands SET LIST ITEM PARAMETER and GET LIST ITEM PARAMETER

SET LIST ITEM PARAMTER ({*; }*list*; *itemRef* | *; *selector*; *value*)

GET LIST ITEM PARAMTER ({*; }*list*; *itemRef* | *; *selector*; *value*)

Parameter	Type	Description
*	*	If omitted (default): use list reference ID for list parameter If specified: use list object name for list parameter
list	ListRef (Longint) | String	Name of the list form object if * specified List reference ID number if * omitted
itemRef or *	Longint | *	Item reference number or 0 for the last item added or * for the current item of the list
selector	String	Parameter constant
value	String | Boolean | Number	Value of the parameter

These commands get and set the value of different parameters based on what is passed into the command as the *selector*.

These are basically developer designed parameters (with one exception) and can be used to store any data of type Text, Number or Boolean. The **SET LIST ITEM PARAMETER** command creates a parameter of the name passed in *selector* (if that selector is being used for the first time) or references a parameter that can have been created by calling the command previously. The type of value passed into the *value* parameter describes what data type will be stored here for all list items using this parameter name. The **GET LIST ITEM PARAMETER** command works similarly, but instead returns the value previously passed using the set command. These commands are powerful because they allow the developer to store many different types of information beyond simply having the text associated with a list item with it.

The exception to the parameters being developer defined is the constant Additional text. The type for values to be passed and retrieved from Additional text are Alpha and anything stored in this parameter is displayed on the right in the Hierarchical List when the item it refers to is selected as the current item of the list.

Command LIST OF CHOICE LISTS

LIST OF CHOICE LISTS (numsArray; namesArray)

Parameter	Type	Description
numsArray	Longint Array	Numbers of choice lists
namesArray	Text Array	Names of choice lists

This command returns two arrays, one an array of the numbers of all choice lists defined by the list editor in Design mode and the other the name. The numbers correspond to the lists' order of creation.

Conclusion

This Technical Note offers a summary of the updated functionality of Hierarchical Lists in 4D v11 SQL. Included is a sample database with many examples of the code needed to implement these new commands and functions in existing or new databases to display different types of data.