“Embedded Controls” for Microsoft Business Solutions

(Technical)

Microsoft Business Solutions

DISCLAIMER

COPYRIGHT NOTICE

TRADEMARKS

Published by Logic Vision B.V., Netherlands.

Published in Netherlands 2003.

DocID: “Embedded Controls MBS (technical)

TABLE OF CONTENTS

Chapte r 1

PREFACE

For a more technical information about “Embedded Control”, this manual is made.

Chapter 1

“Embedded Controls””

1.1 SERVER SIDE INSTALLATION

The server side installation has to be performed on the machine where the Navision server is located. But it is not always necessary to do an installation here. This depends on if you have your Navision license file stored locally on each client or stored on the server.

If your Navision License file is stored on the server the “Embedded Control” can not verify this file so it will always start up in Demo mode. To prevent that the Navision License is to be stored on each client, a “License Server” program is written to allow the “Embedded Control” to remotely access the Navision License File.

To install the “License Server” run the setup.exe” program. When installation options are displayed please manually check the “License Server” option. If the server machine is never used as a client machine you could deselect the “Embedded Controls for Navision product” option, however it is not harmful if you leave this option checked.

When the installation is done the “License Server” program is automatically started. Here you must do a little configuration. Go to the “File” menu and then to settings”.

The Settings form is displayed. In the settings form you must select the location of your company’s license file. Press the “..” button to browse to the location of the Navision license file.

Depending on what IP ranges you use on your network you will have to add a trusted IP range. What does this mean? Standard the “License Server” does only permit license info requests from the standard IP ranges that are reserved for local area networks. If you are not using one of these IP ranges, you must add the IP range(s) that your company uses to the trusted IP’s list. This is to prevent abusive use of the “License Server”.

The standard supported IP ranges are:

“192.168.000.000 – 192.168.255.255”

“010.000.000.000 – 010.255.255.255”

“172.016.000.000 – 172.041.255.255”

“127.000.000.001” (localhost)

If you are using one of these IP ranges you won’t have to configure the trusted IP list.

Adding an IP range can be simply done by clicking the “Add” button. Here you can fill in 2 IP addresses. These IP addresses and every address in between them are now allowed to do License Information requests. But there are some restrictions in adding IP ranges. You can only add ranges where the first 2 series of numbers in every IP address are the same.

If your network works with 192.0.0.x IP numbers, then add the IP range “192.0.000.000 – 192.0.000.255”. The widest range you can use is from “192.100.000.000” to “192.100.255.255”. If you require wider ranges you should add multiple ranges to your “trusted IP’s” list.

If you like, you can change the port that the “License Server” is using if the standard port is not suitable in your situation.

You can minimize the “License Server” and it will still be visible in the system tray. When the lighthouse is lighted the “License Server” is listening for incoming request. If the lighthouse is not lighted then please check your configuration. There may be clues of the possible cause in the message log.

1.2 CLIENT SIDE INSTALLATION

On every Navision client the “Embedded Controls for Navision” product has to be installed. This can also be done by running the setup.exe” program. Installing the “License Server” option has no use on client computers, so you can keep this option unchecked.

1.3 NAVISION OBJECTS INSTALLATION

Only once you have to import some objects into your Navision database. Some of them are required for “Embedded Controls to work, some of them are technical demos for each “Embedded Control” to show most of it’s functionality.

In the installation folder of the “Embedded Controls for Navision” product you will find the file “EC_Demo.fob”. Import this file into your database. The “Embedded Controls” product is now completely installed, however it will work in demo mode.

1.4 “EMBEDDED CONTROL” CONFIGURATION

Though the “Embedded Controls do work with full functionality, a demo message will appear when you close a form containing an “Embedded Control” and a lighthouse will appear with our company logo.

To remove this demo functionality you can obtain an “Embedded Controls License file” by rightclicking the lighthouse in the system tray that appears when an “Embedded Control” is running. Click the menu item “Order” and follow the on screen instructions.

If you have obtained an “Embedded Control” License File” then you must distribute this file on each client computer. When starting up an “Embedded Control” (does not matter which) an lighthouse will appear in the system tray. When right clicking on the lighthouse a menu will appear. Click on the settings” menu and a form will pop-up.

Here you have 2 options to select the “Navision License File”. This is by locating it on the client’s local hard disk or by locating the computer where the “License Server” is installed. So you enter a local filename or a server name (or IP address). Using both options is not possible and of no use.

Second you must specify where the “Embedded Controls License File” is located. This file must be on the clients computer. When the “Embedded Controls License File” matches with the “Navision License File” then no demo messages will appear for the “Embedded Controls” you have purchased.

You can also automate the configuration of “Embedded Controls” on each system. This can be done by writing some C/AL code in a Navision object. To do so, create an OCX variable and reference it to “EC_Manager.Manager”. Type for example the following C/AL code in a proper place:

```
CREATE(<OCX var>);

<OCX var>.SetupEC('192.168.0.1',

                   19000,

                   'c:\fin.flf',

                   'c:\EC_License.ecl');

CLEAR(<OCX var>);
```

Parameter 1 is the IP address or computer name of the “License Server”. When you don’t use the “License Server” you can put in an empty string here.

Parameter 2 is the port where the “License Server” responses to. You can enter ‘0’ when you are not using the “License Server”.

Parameter 3 is the location of the “Navision License File”. You can leave this field blank if you are using the “License Server”.

Parameter 4 is the location of the “Embedded Controls License File”

You can for example execute this code just before an “Embedded Control” is created.

1.5 SETTING UP AN "EMBEDDED CONTROL"

Displaying controls in Navision is rather easy. You just select them from the toolbox and drag them onto a form. Showing an "Embedded Control" does not work that easy. You roughly have to make to steps until the control is visible. The two steps are:

1. Draw a container (SubForm control) on the desired form.

2. Program code that uses that SubForm as container to display the "Embedded Control" in.

The first step is to create a SubForm onto your form. Then give the SubForm a name that represent the control you are using. For example "ChartSpaceContainer". If you will use more then one chart on a form the best thing you can do is to give it a meaningful name of what kind of statistic the chart will represent. For example "ItemSalesChartSpaceContainer".

Set the SubFormID property of the SubForm to the form "Embedded Control" Container". This special form is distributed with the "Embedded Controls product and can host any type of control distributed with the "Embedded Controls product. No additional programming is required in the "Embedded Control Container" form.

The advantage of using a SubForm as control container is that you can use the style properties of the SubForm to give you some extra control over the look of the "Embedded Control".

A very important property you should consider using are the glue properties. These properties allow you to resize the SubForm when the user resizes the parent form. When the SubForm is resized due to the glue properties your "Embedded Control" will also resize. It will keep the same size as the SubForm!

The Border, BorderStyle, BorderColor and BorderWidth properties can also be used to enhance the look of an "Embedded Control". You can use the Enable and Visible property of the SubForm, but only use these properties in code at a point where the "Embedded Control" is already created. Otherwise the creation of the "Embedded Control" will fail.

The second step is to write code that changes the SubForm into the desired "Embedded Control". For this we have to write some C/AL code. To be able to control the "Embedded Control" programmatically we have to create an automation variable that references the

“Embedded Control”. But we do not directly reference the “Embedded Control”. We will reference a DLL that wraps around the “Embedded Control”.

The wrapper DLL will take care of displaying the “Embedded Control” in the SubForm and will also forward all function and properties used in the C/AL language to the “Embedded Control” plus it will forward events (triggers) generated by the “Embedded Control” back to the C/AL language.

Create a global automation variable that references the wrapper DLL. Give this variable a meaningful name like “ChartSpaceControl”. The type of the variable is “Automation”. The subtype of the variable depends on the control you want to use. The chart control is called “navChartSpaceWrapper.CChartSpace”. All control wrappers start with the 3 characters “nav”.

If you want to receive events (triggers) from the control then do not forget to set the “WithEvents”-property of the variable to “Yes”. This property is only available in Navision version 3.00 or higher.

Until now the only thing the SubForm does is showing us an empty form. Somehow we have to draw the control into the SubForm. To achieve this we have to write some code in the OnActivate trigger of the SubForm.

The basic code to display an “Embedded Control” is:

```
IF ISCLEAR(ChartSpaceControl) THEN BEGIN

  CREATE(ChartSpaceControl);

 CurrForm.ChartSpaceContainer.FORM.ControlIsCreated;

  <do some initialising code here>

END;
```

Always us this code like this. Although there will be situations where you won’t need the IF condition, please use the code like this to prevent future problems.

As well as manually creating the control we will have to manually destroy the control. This is done in the “OnCloseForm” trigger. Add the following code in this trigger:

```
ClearAll();
```

Now the work is almost finished. The only thing left to do is to make sure that the “OnActivate” trigger of the SubForm is executed when the form is opened. To achieve this place the following code in the “OnOpenForm” trigger.

```
CurrForm.ChartSpaceContainer.FORM.ActivateMe;
```

Some programmers may notice that there is another way to activate a subform. Well, there is. But somehow this way works different en using the standard ACTIVATE method will cause problems in some cases.

Something to consider is when using multiple tabs on the form and the “Embedded Control” is not on the first tab this code will have the behaviour that the tab where the “Embedded Control” is displayed when opening the form. To avoid this after activating the SubForm activate any control on the first tab. This will result in the first tab being displayed when the form is opened.

At this point all required programming to make the control technically work is done. When running the form the control splash screen will be visible however no chart will be shown. This section can be applied to all controls shipped with the “Embedded Control” module. We can now continue to the functionality of the controls.

1.6 IMPLEMENTING A SPECIFIC ‘“EMBEDDED CONTROL”’

Implementing the Chartspcace control:

Display on forms:

In the previous paragraph we have described to embed a control into your own form. This is an important step for displaying charts. If you have not carried out that procedure yet please do so before continuing with this paragraph.

In this chapter we presume that basic knowledge of Navision forms is available.

How can I present my data with the chart control? First imagine a chart as a 2D matrix that contains a lot of value. An 2D matrix has an X and an Y axis. Learn that the X axis represent the categories and the Y axis represents the Series.

Then we have a category and a serie description array. They can be filled with the SetCategory and SetLegend subroutine. The SetLegend subroutine belongs to the X (horizontal) row in the matrix and the SetCategory subroutine belongs to the Y (vertical) row of the matrix. The following table explains what parameters to use to fill a 3x3 matrix.

SetCell(0,2,…)	SetCell(1,2,…)	SetCell(2,2,…)	**SetLegend(2,…)**
SetCell(0,1,…)	SetCell(1,1,…)	SetCell(2,1,…)	**SetLegend(1,…)**
SetCell(0,0,…)	SetCell(1,0,…)	SetCell(2,0,…)	**SetLegend(0,…)**
SetCategory(0,…)	**SetCategory(1,…)**	**SetCategory(2,…)**	**<X ^Y**

Now the trick is to fill the Matrix in a way that is most efficient. This depends on what data you want to present and how you want the data to be presented.

Stating the above there is no general way to enter data in a graph. That’s why we have created an interface that let’s you decide if you want for example enter data by categories or by series.

The method to do this is the SetCell method. This instruction allows you to enter the X and Y coordinates of the matrix and enter a value in it.

With this system it is not required to enter the first item in the matrix first. You can decide in what order the data will be entered into the matrix. You can also adjust one item in the matrix after it has been set.

Is there any limit to the number of series and categories I can put in the matrix? Yes there is a maximum of 256 (0-255) to enter as series and as categories. If you enter higher values there will not occur any error but it just doesn't have effect. Anyway the human readable maximum is much lower for 3D charts. 2D charts can display large data better. So unless you have a 30" display this maximum is enough.

An example of a routine that goes for most solutions:

(define 3 local variables first, X = integer, Y = integer, Text = Text)

```
FOR X := 0 TO 10 DO

BEGIN

  Text := 'Category ' + FORMAT(X);

  ChartSpaceControl.SetCategory(X,Text);

  FOR Y := 0 TO 5 DO

  BEGIN

    IF X = 0 THEN

    BEGIN

      Text := 'Legend ' + FORMAT(Y);

      ChartSpaceControl.SetLegend(Y,Text);

    END;

    ChartSpaceControl.SetCell(X,Y,RANDOM(100));

  END;

END;

ChartSpaceControl.Type := 17;

ChartSpaceControl.Show;
```

This little routine does 5 things.

1. It puts data in the matrix
2. It fills the category names
3. It fills the Series names
4. Sets the chart type to type 17
5. Finally it displays the chart.

The data we see is not very meaningful as it is Random. Within the loops you can collect your data from any table you want and use the SetCell instruction to enter this data into the matrix/chart.

You could also switch the X loop and Y loop so you can fill the chart by serie.

In addition here is a small routine that really brings your chart alive:

(put this in the OnTimer trigger and set it’s interval at 100)

```
// adjust 3 cells in the matrix...

ChartSpaceControl.SetCell(RANDOM(10),

                              RANDOM(5),

                              RANDOM(100));

ChartSpaceControl.SetCell(RANDOM(10),

                              RANDOM(10),

                              RANDOM(100));

ChartSpaceControl.SetCell(RANDOM(10),

                              RANDOM(5),

                              RANDOM(100));



// rotate the chart...
```

```
ChartSpaceControl.Rotation :=
           ChartSpaceControl.Rotation + 1;

IF ChartSpaceControl.Rotation => 360 THEN
  ChartSpaceControl.Rotation :=
    ChartSpaceControl.Rotation - 360;
```

Display on reports:

Displaying charts on reports are a little more complicated then displaying charts on a form. However it’s not that bad but I recommend that you go through the “Display on Forms” chapter first. Because this chapter doesn’t explain the basic interface anymore.

In this chapter we presume that basic knowledge of reports is available.

To create charts on the fly in reports we will need a control container for the chart in order to let the system work. This control container does not have to be the actual size of the chart on paper. Later the ExportPicture function will render the chart into the required sized. No stretching is done when the chart on the form is smaller, the chart will be completely redrawn to properly fit on the report.

This control container needs a place to sit on. The only place for this will be the Request form. The chart will appear on the options form, but this is nice because there you can allow the user to preset some settings such as rotation etc.. These settings then will also apply to the actual print unless you overrule them witch code.

To create this container onto the SubForm follow the procedure described in the setting up an “Embedded Control” section. There however is a small difference in the programming. We don’t activate the SubForm in the “OnOpenForm” trigger but in the “OnActivateForm” trigger.

We have to do this because when activating the SubForm when opening the report this will result in the “Options” tab to be displayed first. Many times this is not what we want. There is no way to go back to the first tab using C/AL coding. So the only way is to use the “OnActivateForm” trigger (which will trigger when clicking on the “Options” tab).

But then we have to solve another problem. The user might not click on the Options tab. This would result in an error running the report is printed or previewed because the automation variable is then never created. So the first thing to do when printing or previewing is to create the automation variable if it has not jet been created. This is done in the OnPreDataItem secton. Here we can use the same code as in the “OnActivateForm”.

```
RequestOptionsForm.ChartSpaceContainer.FORM.Activate
Me;
```

The “OnActivate” trigger of the SubForm will be triggered. When the “Embedded Control” is not yet created the “Embedded Control” will be created. If it is created then nothing will happen.

In the section designer create the sections and items required. Make a global variable to the “Picture Container” record. Set the property “Temporary” to “YES”. This is very important to avoid that BLOB chart objects are written and stored into the database. Now the charts will only exist on the MBS Navision client side.

Now in the section where you want your chart create a PictureBox control. Set it’s source expression property to the “Blob Field” of the “Picture Container” record. Draw the PictureBox in any size you want. It can be big, because the ExportPicture subroutine doesn’t stretch the chart when the original on the request form is smaller, but it renders it at the given size so the picture will always be of the proper resolution.

In the OnPreSection insert the following code:

```
ChartSpaceControl.Clear;

ChartSpaceControl.Type := 17;

// ... ... ... ... ... ... ... ... ... ... ... ... .

// INSERT YOUR CODE HERE TO ENTER DATA IN THE CHART

// ... ... ... ... ... ... ... ... ... ... ... ... .

ChartSpaceControl.Show;

// Create new entry in temporary "Picture Container"
record.

PictureContainer.INIT;

IF PictureContainer.FIND('+') THEN
```

```
  PictureContainer.Entry := PictureContainer.Entry +
1;

ELSE

  PictureContainer.Entry := 0;

PictureContainer.INSERT;



// Export the chart to file and load it in the
"Picture

// Container" BLOB record.

ChartSpaceControl.ExportPicture('c:\Graph.bmp',18150
,9729);

PictureContainer."Blob
Field".IMPORT('c:\Graph.bmp');

ChartSpaceControl.DeletePicture('c:\Graph.bmp');
```

Most of the routine is normal C/SIDE programming or is covered in previous chapters. But the last 3 lines may need a little explanation.

The ExportPicture subroutine saves the ChartSpace object to disk. Therefor it requires a filename, a picture width and pictureheight. How do I know the picture width en picture height…? That is easy. Go to your PictureBox control and look at it’s width and height properties. Insert those values as arguments to the ExportPicture subroutine and it produces a chart picture that is rendered to fit exactly in your PictureBox control.

After the export we need to import the bitmap file into a blob. This blob is contained by the “Picture Container” record.

After the import we can safely delete the bitmap file. It will not be used anymore by Navision after the import. Deleting can be done by the DeletePicture subroutine. In fact this subroutine could delete any file.

In general the fastest way to create charts in reports is to build the chart on a form. When the code works fine and your chart looks terrific then you can copy the code into the part marked as “insert code here to enter data in the chart” in the example code.

Implementing the PictureViewer Control

For now the PictureViewer control only works for displaying on forms. Printing in reports will require the old fashioned conversion to BMP routine.

The PictureViewer control is a control that can display many file formats onto a form without importing it into a BLOB. This solves a much big shortcoming of Navision. Navision can only display Bitmap files.

The Graphics formats that are supported by the PictureViewer control are:

- BMP (bitmap)
- ICO (icon)
- CUR (cursor)
- RLE (run-length encoded)
- WMF (metafile)
- EMF (enhanced metafile)
- GIF
- JPG

After doing the standard procedure described in the chapter setting up an “Embedded Control” most of the work is already done. Down here is a simple procedure that utilises all properties:

```
PictureViewerControl.FileName := ‘c:\train.jpg’;

PictureViewerControl.RenderStyle := 2;

PictureViewerControl.EnableZoom := TRUE;

PictureViewerControl.ZoomLevel := 1.5;

PictureViewerControl.BackColor := 14215660;
```

The first property is very straight foreward. It simply loads and displays the picture you specify.

The RenderStyle property defines the way you initially want your picture to be displayed. The possibilities are:

0. FullSize

1. Fit

2. FitKeepProportion

RenderStyle 0 displays the picture in it's original size. RenderStyle 1 will always keep the picture stretched exactly in the control. RenderStyle 2 will make the picture as big as possible to fit in the control but keeps the pictures proportion. When zooming in and out the fitting is no longer applied unless you set the RenderStyle again.

The EnableZoom property allows the user to zoom in and out. When this property is set to TRUE and the user is holding the mouse over the picture, the mouse cursor changes into a zoom-in and zoom-out button. When the user clicks the left button the picture will zoom in and when the user clicks the right button the picture will zoom out. This property has no effect when the RenderStyle is set to 0.

With the ZoomLevel property you can set a ZoomLevel by code. The EnableZoom property does not have to be set to TRUE to let this property work. But it only works with RenderStyle 0 and 2.

Whenever the picture exceeds the size of the control there will appear a horizontal and/or vertical scrollbar to allow the user to scroll the picture.

Sometimes there will be edges above or at the side of the control. These can be given a color by the property BackColor. If you want it to have the same color as your form use the form's BackColor property number to set the control's background color.

Tip: Use the BorderStyle property of the SubForm to change the look of any control.

Implementing the Slider Control

The slider control is designed to control a 3D chart a little bit more easy. But the slider control could serve many more purposes. I'm sure many C/AL programmers and application designers at a time wished to have some additional controls that made specific tasks a little bit more easy.

In addition to the ChartSpace and PictureViewer control he slider control has an extra special feature. This is that it fires events (or triggers) back to Navision. This allows the C/AL programmer to respond easily and immediately when the user is sliding the slider.

When defining an automation variable to navSliderWrapper.CSlider be sure to set the "WithEvents" property to "Yes". Unfortunately this property is only available from Navision 3.0. In earlier version of Navision this functionality of the control isn't there. Therefore in these versions the slider control becomes pretty much useless. It will show, it will slide, but it won't trigger C/AL code. The only way of getting some interaction is by using the timer.

```
SliderControl.Init(0,100,5,1,50,0);
```

This code does the same as:

```
SliderControl.MinValue = 0;

SliderControl.MaxValue = 100;

SliderControl.ScaleStep = 5;

SliderControl.SlideStep = 1;

SliderControl.Value = 50;

SliderControl.Orientation = 0;
```

The fist example is better because the control will show up more smoothly, because it can draw itself in only once and will avoid screen distortion.

With the MinValue and MaxValue property you can set the boundaries of the Value property. The Value property holds the current value of the slider. The ScaleStep property is used to set the distance of the scale lines of the slider. With the SlideStep you can step over numbers when the user slides. You can use the orientation property to create an horizontal (0) slider or an vertical (1) slider.

Some additional properties of the slider control are:

```
SliderControl.BackColor = 0;

SliderControl.KnobColor = 14215660;
```

With these 2 properties you can change the looks of the slider control. The values range from 0 - 16,777,215. You can check get the number of the color you like by changing a navision form color and use the backcolor property value of that form as parameter.

Of course you can make the slider control in any size you want depending on how large you make the SubForm that contains the slider control.

1.7 TROUBLESHOOTING FAQ

Q: When I try to make a customized "Embedded Control" on my form the control isn't visible. What am I doing wrong?

A: Please follow up the steps in chapter 3 carefully. You should find an answer here. If you get error messages then be sure you have installed the wrapper, the OCX and when you are using the ChartSpace "Microsoft Web Chart 10.0" has to be installed (read the sytem requirements" part of the "Product FAQ").

Check the C/AL debugger if the CREATE() method is executed

Q: I get an error message when starting one of the demo objects. What do I need to do to fix this?

A: Please reinstall the product and then it should work fine. The demo objects work from Navision 3.00.

Q: I' am not getting the expected results when changing properties or calling subroutines in the control. How do I find out what is going wrong?

A: The "Embedded Controls" are designed to give a minimum of error messages. Almost all error messages are send to one single logging module. You can find them by right clicking on the lighthouse in the system tray and select the menu item "Error Log". The form that opens here should not give any error message (unless you are working in demo mode).

You may find an entry in this error log that explains a little bit of what went wrong. A dump of the parameters that were used in the function/property that reported the error is also made in this error log.

Q: Why am I not having event triggers in C/AL after declaring an automation variable?

A: First of all the "WithEvents" property of the automation variable must be set to "Yes". You can find this property in the "C/AL globals" menu, select the automation variable and open the properties dialog. Secondly you should know that the only "Embedded Control" that contains event triggers is the "Embedded Slider Control". With all other controls no triggers will appear.

Q: Navision 2.60 crashes when using the Demo objects. How do I solve this?

A: Although it is possible to use “Embedded Controls” in Navision 2.60, it needs a different approach. Navision 2.60 is limited in some ways so the “ChartSpace Settings – TechDemo” form will not work. The slider won’t work either because Navision 2.60 does not support events.

The best thing you can do is adding an global Boolean variable that you set to true after using CREATE when you create the automation variable. Use this Boolean variable as the condition to create the automation variable instead of the function “ISCLEAR” (doesn’t exist in 2.60). Remove all other code in the “OnActivate” trigger of the SubForm control.

1.8 PRODUCT FAQ

Q: What is the advantage of embedding controls in Navision instead of using for example the functionality of Excel?

A: Embedding means that the control really is displayed on a Navision form. This means that the end user does not has to switch to excel or another application that provide additional functionality. And Because the controls are controlled by C/AL code you can easily and quickly write code in the OnAfterGetRecord trigger so that the control is instantly updated when browsing through records in Navision.

Q: What are the system requirements to use the control?

A: There are some things need to be taken in consideration.

- The “Microsoft Web Components 10.0” control has to be installed on the client machine. When you have a new Microsoft Office version (XP or higher) then this product is most likely already installed on your system. If you don’t have this product, please download the “Microsoft Office XP Web Components” from this locations (it’s for free):

  http://www.microsoft.com/downloads/details.aspx?FamilyID=982B0359-0A86-4FB2-A7EE-5F3A499515DD&displaylang=en

- This product is designed to be used with Navision. It can be used in any version of Navision version that supports automation. The Slider control however is limited to Navision version 3.00 or higher. This document only describes the use of the product from Navision 3.00. For lower Navision versions adjustments in C/AL coding have to be made.

Q: I have made my own control. Can I use the wrapper DLL to display my control embedded in Navision?

A: No, this is not possible. The wrapper DLL is designed to support the interfaces of the controls that are shipped with the “Embedded Control” package. Therefore for each additional control the wrapper DLL has to be adjusted to support this control’s interface.

The only way you can do this is by writing your own wrapper DLL.

Q: The ChartSpace control looks nice. But we want the chart to be printed in Navision reports. Is there a way to do this?

A: Yes, there is. You can create a chart in a report on the fly and export it to a bitmap using a resolution that best fits into the PictureBox. After the chart is exported just import it into a temporary BLOB record that is in the source expression of the PictureBox. See the manual for more information. There is a demo in the fob that demonstrates the usage of the Embedded ChartSpace control.

Q: I like your controls. Is it possible to use these controls in different development environments then Navision?

A: Yes you could use the OCX in most development environments however many development environments already support displaying custom controls so there is no need to use our OCX as windows and office already provide these controls. Then you could use full functionalities of the controls. Anyway you can use the OCX without the wrapper DLL or with the wrapper DLL.

This product is made for use with Navision, but I believe for other development environments there are far better solutions available that are better suitable for that development enviroment. I'd recommend that you use these.

Q: Is your company logo removed from the controls when I buy this product.

A: Of course, The full space of the Form will be available to the control. Our company logo will not pop-up. Also the message that this product is a demo version will not show up. That's why we hope you would have a reason to (ugh) buy this product…

Q: I have seen the demo and I love it. Where can I buy this?

A: See the lighthouse in the systemtray when one of the “Embedded Controls is running? Right click on it en select “order”. Follow the instructions to order the “Embedded Controls Product.