OMNIS Web Client

April 1999

Table of Contents

ABOUT THIS MANUAL **6**

CHAPTER 1–INTRODUCTION **7**

THE CLIENT 9
- *Web Browser* *9*
- *OMNIS Web Client* *9*
- *Installing the Web Client for Development* *10*

THE SERVER 10
- *The Web Server* *10*
- *The Web Server Extension* *11*
- *The OMNIS Runtime* *11*
- *OMNIS Application* *12*

STANDARD HTML FORMS 12

CHAPTER 2–OMNIS WEB CLIENT TUTORIAL **13**

DESIGNING THE REMOTE FORM 14
- *Using the Sidebar Remote Form Wizard* *15*
- *Editing Form Fields* *27*
- *Adding Variables and Methods to the Form* *28*
- *Events and Adding Methods to Form Fields* *33*
- *Adding a Picture Field* *34*
- *Testing the Form and Browsing the Books* *36*
- *Debugging the Form* *36*

DEPLOYING THE REMOTE FORM 37
- *OMNIS Server* *37*
- *Web Server and HTML files* *38*

VIEWING THE OMNIS WEB CLIENT EXAMPLE LIBRARY 40

CHAPTER 3–DESIGNING OMNIS REMOTE FORMS **41**

CREATING REMOTE FORMS USING WIZARDS 41
- *Plain Form Wizard* *43*
- *Tabs Form Wizard* *44*
- *Wizard Form Wizard* *49*
- *Submit Form Wizard* *50*
- *Password Form Wizard* *52*
- *Sidebar Form Wizard* *54*
- *Sidebar with pages Form Wizard* *57*
- *Multiform Form Wizard* *58*

DEBUGGING AND TESTING REMOTE FORMS 61

CREATING A NEW REMOTE FORM 61
REMOTE FORM PROPERTIES 61
Icon Pages *61*
REMOTE FORM CONTROLS 62
Labels *63*
Border Objects *63*
Pushbuttons *63*
Single Line Entry Fields *63*
Date and Time Display Format 64
Multi Line Entry Fields *64*
Check Boxes *64*
Radio Group Objects *64*
List Boxes *64*
Drop Lists *65*
Combo Boxes *65*
Headed Lists *65*
Picture Objects *65*
Sidebars *65*
Tab Bars *66*
Page Panes *66*
PROGRAMMING REMOTE FORMS 67
Optimizing Data Handling *67*
Icon pages *68*
Open windows and User Prompts *68*
OK Messages 68
Smart lists *69*
REMOTE FORM INSTANCES 69
FORM CACHING 69
Server Form Caching *69*
Client Form Caching *69*
EVENTS 70

CHAPTER 4–REMOTE TASK CLASSES 72

CREATING REMOTE TASK CLASSES USING WIZARDS 72
Plain Remote Task Wizard *73*
Monitor Remote Task Wizard *74*
Monitor Window 75
HTML Report Task Wizard *76*
Submit Remote Task Wizard *77*
CREATING A NEW REMOTE TASK CLASS 78
REMOTE TASK INSTANCES 78
EVENTS 79
CLIENT ACCESS STATISTICS 79
CONNECTING USING STANDARD HTML FORMS 80

Task Wizards and $construct *83*

CHAPTER 5–DEPLOYING YOUR OMNIS WEB SOLUTION 84

EDITING YOUR HTML PAGES 85
Embedding the ActiveX Web Client *85*
Embedding the Netscape Plug-in Web Client *87*
CGI *87*
OMNIS SERVER CONFIGURATION 88
Setting the OMNIS Port Number *88*
NT Servers *89*
WEB SERVER CONFIGURATION 89
Installing the Web Server Extension *89*
ISP Web Hosting *89*
Secure Sockets *90*
OMNIS WEB CLIENT INSTALLERS 91
Creating your own Installers *91*
Web Client Components *91*
Remote Form Components *91*

CHAPTER 6–TROUBLESHOOTING GUIDE AND FAQ 93

GENERAL 93
REMOTE FORMS 93
BROWSER 94
OMNIS SERVER 94

CHAPTER 7–REFERENCE 96

EVENTS 96
Remote Task Classes *96*
Remote Form Objects *96*
NOTATION 97
OMNIS Preferences *97*
Remote Form Class *98*
Remote Form Objects *100*
Remote Task Class *102*
Remote Task Instance *103*

About This Manual

This manual describes the OMNIS Web Client™ and how you use OMNIS Studio to develop applications and business solutions that you can deploy to the World Wide Web.

This manual contains the following chapters:

– **Introduction**
  presents an overview of the OMNIS Web Client technology, and describes the different components required in your complete OMNIS web solution.

– **OMNIS Web Client Tutorial**
  takes you through creating a simple application that browses a Bookstore datafile, and describes how you can deploy this to the web.

– **Designing OMNIS Remote forms**
  describes how you create OMNIS remote forms, the GUI classes that the user accesses in their browser; this chapter includes a description of the different remote form templates and wizards available in OMNIS Studio.

– **Remote Task Classes**
  describes how you use remote tasks in association with remote forms, and includes a description of the different remote task templates and wizards

– **Deploying Your OMNIS Web Solution**
  describes how you set up your html pages containing the OMNIS Web Client, and shows you how to set up the server-side of your web solution.

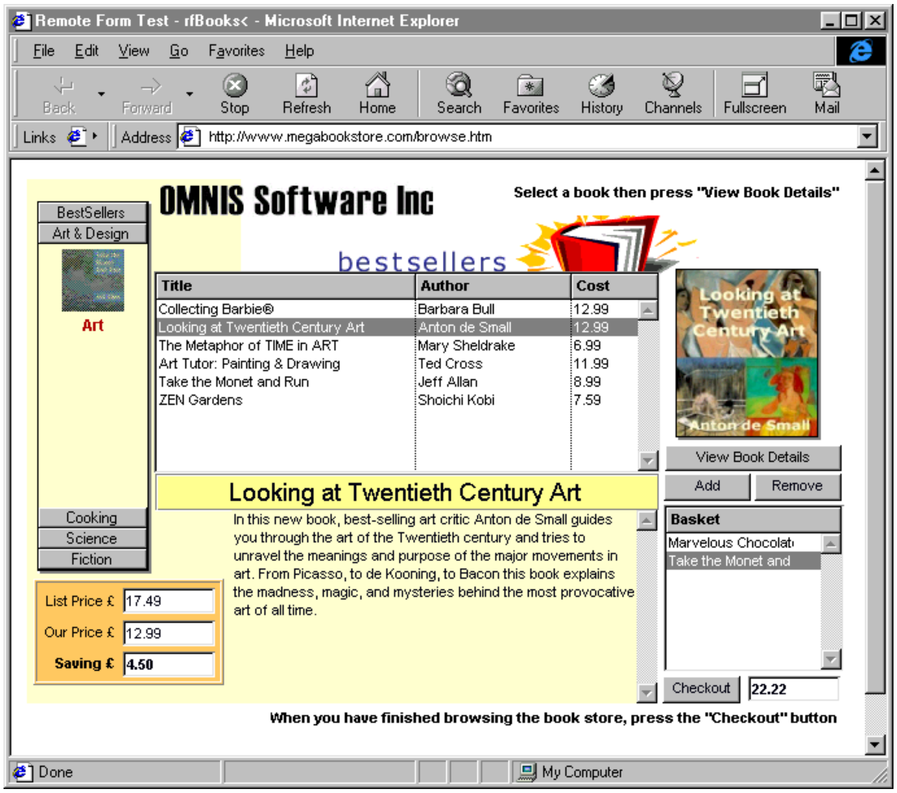
– **Troubleshooting Guide and FAQ**
  answers the commonest questions and problems concerning remote forms and the OMNIS Web Client

– **Reference**
  Lists all the events, properties and methods associated with remote forms and tasks.

Chapter 1–Introduction

Using OMNIS Studio and the OMNIS Web Client™ you can create any type of application and deploy it to the web. The application, written in OMNIS Studio, can do anything from healthcare management to multi-media presentations and publishing, to human resources, to electronic commerce, to education or government administration. The OMNIS Web Client lets you embed your application into a standard web page, and allows anyone with a suitable web browser, anywhere in the world, to access your data 24 hours a day, 7 days a week. And what makes this even more incredible is that the forms you build in your application are created entirely using OMNIS Studio's own 4GL, wizards, and web components, no Java and no complex html is required: it's that simple!

The following screenshot shows a web browser containing an example application for browsing books in an on-line bookstore. The form, displayed in the browser using the OMNIS Web Client, contains many types of component including Sidebar, various list controls, form fields, buttons, check boxes, and so on.


Remote Form Test - rfBooks< - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail
Links Address
OMNIS Software Inc
bestsellers
BestSellers
Art & Design
Art
Cooking
Science
Fiction
View Book Details
Add
Remove
Checkout
Done
My Computer

http://www.megabookstore.com/browse.htm

Select a book then press "View Book Details"

Title	Author	Cost
Collecting Barbie®	Barbara Bull	12.99
Looking at Twentieth Century Art	Anton de Small	12.99
The Metaphor of TIME in ART	Mary Sheldrake	6.99
Art Tutor: Painting & Drawing	Ted Cross	11.99
Take the Monet and Run	Jeff Allan	8.99
ZEN Gardens	Shoichi Kobi	7.59

Looking at Twentieth Century Art

In this new book, best-selling art critic Anton de Small guides you through the art of the Twentieth century and tries to unravel the meanings and purpose of the major movements in art. From Picasso, to de Kooning, to Bacon this book explains the madness, magic, and mysteries behind the most provocative art of all time.

List Price £	17.49
Our Price £	12.99
Saving £	**4.50**

Basket
Marvelous Chocolat
Take the Monet and

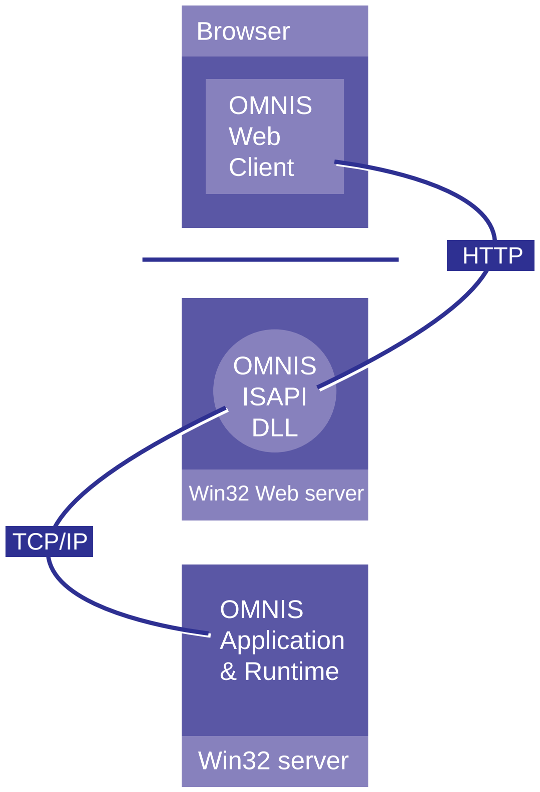
Checkout 22.22

When you have finished browsing the book store, press the "Checkout" button

OMNIS Studio lets you design remote forms that users can access and display in a web browser. A *remote form* is an OMNIS class that contains all the necessary controls to display your data and allow users to interact with your application and database via the web. Once connected an OMNIS remote form posts events to the OMNIS server application, which in turn sends results back to the client.

You can adapt an existing OMNIS Studio application for web access by adding remote forms to your OMNIS application, and adding one or two components to your existing web server and network infra-structure.

An OMNIS Web Client solution has two main parts: *the Client* and *the Server*.



– **The Client**
  The client is the end-user's web browser which they use to access and view your web pages (written in standard html) containing the OMNIS Web Client. The client is embedded in an html page which in turn displays the remote form(s) stored in your OMNIS library located on your server.

– **The Server**
  The server side of your OMNIS web solution comprises your web server, and the OMNIS server, that is, your OMNIS application and the OMNIS runtime engine, and your database. Your web server requires a server extension (such as OMNISAPI.DLL), an invisible component that mediates between the client and the server.

The Client

To access your OMNIS web solution at any time, from anywhere in the world, the user only requires a standard web browser. The first time a user accesses your application they need to download and install the OMNIS Web Client which you can provide on your website. Having installed the client, the user would navigate to the appropriate page(s) on your website containing your OMNIS web solution. This section summarizes the components required on the client machine.

Web Browser

The OMNIS Web Client is available as either an ActiveX or Netscape plug-in, so the client's web browser must support one or other of these plug-ins. For example, users can access a page containing the ActiveX Web Client using Windows Internet Explorer™, or the Netscape plug-in using Netscape Navigator™. When you deploy your application to the web you need to take account of the different browsers in use, and adapt your html pages containing the OMNIS Web Client accordingly.

OMNIS Web Client

The OMNIS Web Client is an ActiveX or Netscape plug-in that you can place in an html page. You could however use the OMNIS Web Client in any development environment that supports ActiveX, including OMNIS itself and many other development environments.
On a Windows client machine, the OMNIS Web Client uses the Wininet API to communicate with the OMNIS web server extension via HTTP, or HTTPS if secure transmission is required. Communications between the OMNIS web server extension and the OMNIS Runtime and library is via TCP/IP.

To use an OMNIS web solution containing the OMNIS Web Client, a user must download and install the OMNIS Web Client. This is a one-time operation. You can provide a link on your website to allow the user to download the OMNIS Web Client installer. Installers for the OMNIS Web Client are provided on the OMNIS Studio CD, and you can make your own custom installers containing the components needed for your application. The latest OMNIS Web Client installers are also freely available on the OMNIS website.

When the user accesses your OMNIS web solution in their browser, the OMNIS Web Client is activated and a connection is established with the server extension on your web server (see The Server section below). Once a connection is established, the OMNIS Web Client receives the class data of the remote form plus all the instance variable data, icon pages, font tables, etc, which it gets from your OMNIS library. The OMNIS Web Client instantiates the visual instance of the remote form and all its controls, enter data mode is initiated, and the user can now use the form.

You can monitor and control what happens in the form by monitoring the *events* triggered by the user while using the form. For example, an event is generated in the form when the user clicks on a form field, when they select a line in a list, press a button, or tab from field to field. You can detect and respond to an event by creating an *event handling method,* stored with the object in your remote form. When an event is triggered, it is sent from the client to the server application and the event handling method for the object is executed. The method could do anything you want, for example, it could save the data in the form to a database, or return the result of a calculation to the client. By default all events are turned off in a remote form to cut down needless traffic, but you can enable specific events for each form control. When you design your application you need to consider what impact the traffic generated by events will have on performance. Details about handling events and tips for reducing network traffic are described later in this manual.

Under Windows, note that the OMNIS Web Client could potentially be placed into any environment which supports ActiveX, including applications written in Delphi or other development tools. The OMNIS Web Client ActiveX could be placed inside a standard OMNIS window.

Installing the Web Client for Development

You must install the OMNIS Web Client to design and test your remote forms in OMNIS Studio. You can test remote forms while you're designing them using Ctrl/Cmnd-T, but this will only work if you have installed the OMNIS Web Client. There is an installer for the OMNIS Web Client on the OMNIS Studio CD in the Webclient/Client folder. There is a separate installer for the ActiveX and Netscape plug-in which installs the Web Client ready for you to start designing and testing your remote forms.

The Server

The server side of your OMNIS web solution comprises a web server, a web server extension supplied by OMNIS, the OMNIS runtime or executable, your OMNIS application, and your data source, either an OMNIS database or proprietary server database. All these various server pieces may or may not be located on the same machine, but typically they would be on the same LAN. This section summarizes the components you need on the server side of your OMNIS web solution.

The Web Server

The OMNIS Web Client requires a stand-alone, industry-standard Win32 web server. Web server software is available from many different third-parties, including web servers downloaded from the Internet. The web server manages the html pages containing the OMNIS Web Client, giving access to your OMNIS application. Using your favorite search engine on the Internet, you can search for "web server software" to find many hundreds of sources for a suitable web server.

Your web server also requires a web server extension, supplied by OMNIS, that handles communications between the client and your OMNIS application. See the next section.

Note that the OMNIS server or runtime executable ***is not a web server***. The OMNIS runtime runs your application, contains the event handling methods, executes the business logic, and so on, it does not serve up html pages. Your OMNIS web solution requires a web server, as described above.

The Web Server Extension

The web server extension is a piece of software supplied by OMNIS that sits on your web server, and listens for and queues any requests from the OMNIS Web Client in the user's browser. Similarly, it passes any results sent from the OMNIS server back to the OMNIS Web Client in the user's browser.

The web server extension is typically installed inside the cgi-bin folder of your web server. At present the OMNIS Web Client supports MS ISAPI and CGI, and a separate server extension is supplied for each interface. In the OMNIS Studio installed tree you can find the extensions under the OMNIS/Webclient/Server/Webserver folder. The ISAPI folder contains OMNISAPI.DLL which conforms to the MS ISAPI interface. The CGI folder contains nph-omniscgi.exe which supports cgi. In future other interfaces may be supported, such as Netscape Server.

If your web server does not support the MS ISAPI interface you will have to use the nph-omniscgi.exe cgi program.

The OMNIS Runtime

To complete your web solution the server-side requires the OMNIS Server or OMNIS Runtime executable, and your OMNIS library containing your remote forms and any necessary business logic and reporting capabilities. The OMNIS Runtime and your library reside on a Win32 server, or a Win32 machine local to your web server.

The OMNIS runtime must be serialized for a specific number of concurrent users. For example, if your OMNIS runtime is serialized with a 100 user serial number any number of people can access your application, but only 100 users are allowed a connection at any one time. The serial number required for the OMNIS Web Client is a special web serial number containing a "W", a standard multi-user serial number containing an "M" will not work in this case.

OMNIS Application

Your remote forms and any necessary business logic for your web solution are stored in an OMNIS library file. Your remote forms are stored as OMNIS *remote form classes*. In addition you need to create a *remote task class* to handle the connection to each client. These classes are described in more detail later in this manual. Designing OMNIS libraries is described in detail in the *Using OMNIS Studio* manual.

OMNIS remote form classes are designed using the OMNIS Studio development environment. You can debug or test your remote forms during development using your own web browser. There are many types of fields and controls you can use in your remote forms. The form controls are implemented as external components, and include pushbuttons, single and multi-line edit fields, radio buttons, check boxes, sidebars, list boxes, drop lists, and combo boxes. In addition, you can use the built-in page pane control in your remote forms. You can create your own controls as external components (using C++) and use them in your forms. The remote form class and its controls are entirely cross-platform, that is, Windows and Macintosh web browsers can access the same forms stored in your OMNIS library.

Note that OMNIS Studio includes a Web Client example library in the Examples Browser, available under the **Tools** menu on the main OMNIS menubar. This example library shows a Bookstore and a Schools administration system using remote forms.

Standard HTML Forms

As an alternative to using the OMNIS Web Client and remote forms, OMNIS Studio lets you interact with your OMNIS application and database over the Internet using standard html forms. In this case, your html forms connect to an OMNIS remote task class direct, and no remote forms are required for this type of interaction. See the Remote Tasks chapter for more details about creating standard html forms for direct access.

Chapter 2–OMNIS Web Client Tutorial

Developing an OMNIS web solution using the OMNIS Web Client is relatively easy once you have learnt how to develop OMNIS applications in general. The first part of this chapter describes how you can create a simple remote form to browse the Books data supplied in the Examples folder in OMNIS Studio. It uses one of the remote form wizards as a starting point for your form and involves some programming. You can create the Books remote form and test it locally in the development version of OMNIS. The second part of the tutorial describes how you can deploy the Book browser window, and assumes you have a personal web server such as the one supplied with Microsoft FrontPage, and it uses the development version of OMNIS to run the application. However for the final deployment stage in your own OMNIS web solution, you will need to purchase and install a stand-alone web server, set up a Win32 server to run the OMNIS runtime and your library, and you'll need to purchase the appropriate license for the OMNIS runtime server.

IMPORTANT: You must install the OMNIS Web Client to do the tutorial, or design and test any remote form for that matter. There is an installer for the OMNIS Web Client on the OMNIS Studio CD in the Webclient/Client folder. There is a separate installer for the ActiveX, suitable for displaying your forms in Microsoft Internet Explorer, and a Netscape plug-in suitable for Netscape Navigator. You can use either the ActiveX or Netscape plug-in for the tutorial, but you must run the appropriate installer before you begin the tutorial.

This tutorial assumes you know how to use the OMNIS Component Store and Property Manager to create and modify fields and controls, and it also assumes you know how to use the method editor to enter methods and variables. For further details about all these topics, see the *Using OMNIS Studio* manual.

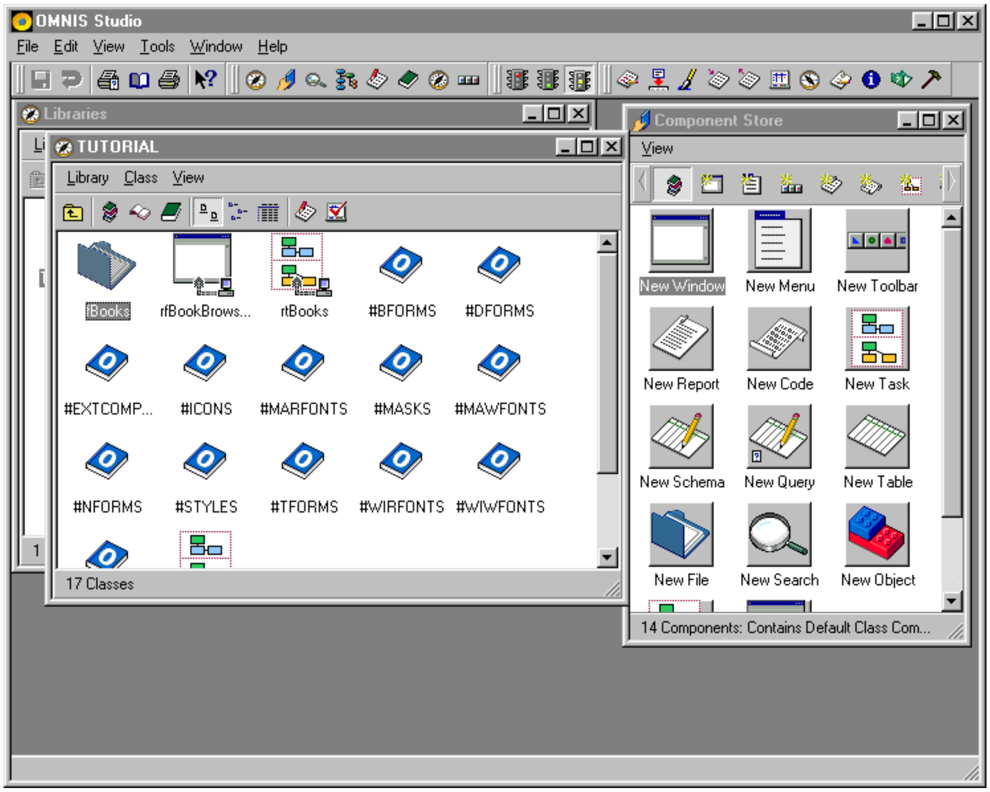
Designing the Remote Form

To develop an OMNIS web solution you need to create an OMNIS remote form using one of the wizards supplied in OMNIS. First, open the Tutorial library:

- Start OMNIS and press Ctrl/Cmnd-O to open the library
- Open the Welcome folder in the main OMNIS Studio folder
- Open the Tutorial/Webclient folder and double-click on the Tutorial.lbs library

The library opens a sample datafile called Webclient.df1 containing the Books data. If the datafile cannot be found, the library will prompt you to open the Webclient.df1 datafile, which is located in the same folder as the Tutorial.lbs library.

- Double-click on the library in the Class Browser to show all its classes

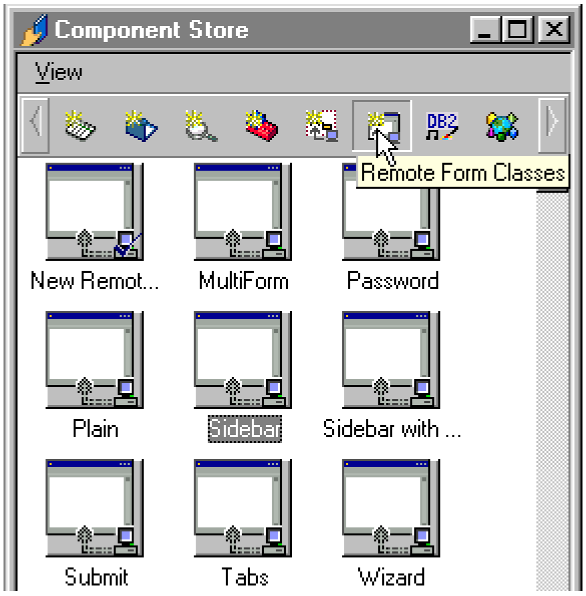


The library contains a number of classes including the **fBooks** file class, the **rtBooks** remote task class, and the **#ICONS** system table containing the necessary icons. The library also contains a remote form class called **rfBookBrowserFinal** which is a backup copy of the form you create in this tutorial. The **Startup_Task** class contains code that is executed when the library is opened, which in this case opens the Webclient example datafile.

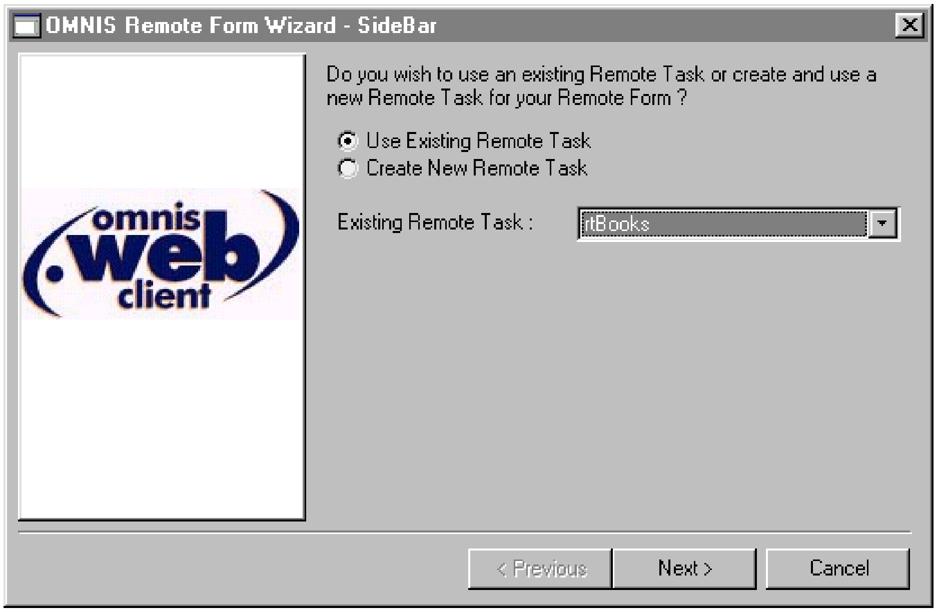
Using the Sidebar Remote Form Wizard

First you need to open the Component Store and view the remote form class wizards.

- Press F3/Cmnd-3 to open the Component Store or bring it to the top
- Scroll the top toolbar in the Component Store and click on the Remote Form Classes button to show the remote form wizards and templates



- Drag the **Sidebar** wizard onto the Tutorial library in the Class Browser (do not use the 'Sidebar with pages' wizard)
- Name the class **rfBookBrowser** and press Return


OMNIS Remote Form Wizard - SideBar

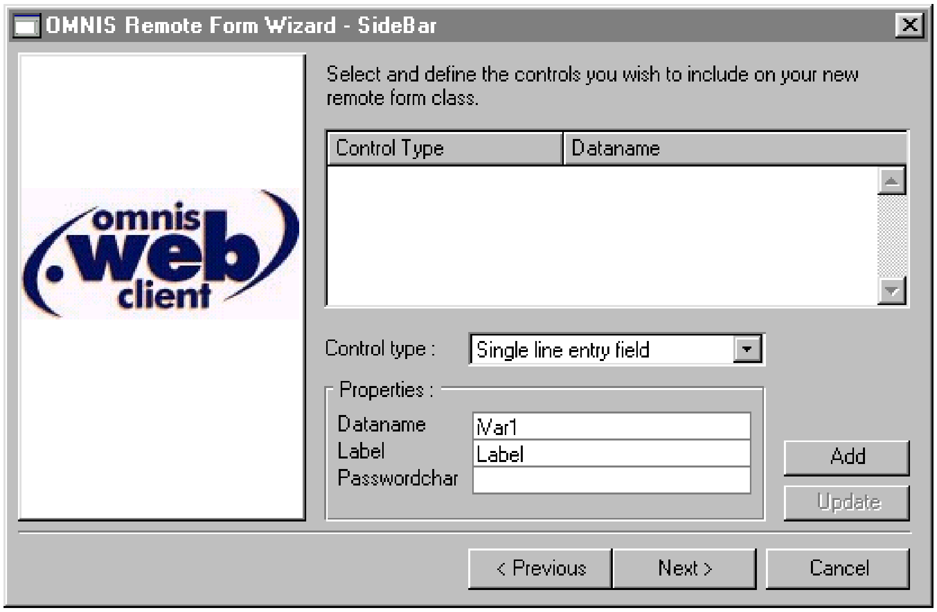
Do you wish to use an existing Remote Task or create and use a new Remote Task for your Remote Form ?

[✓ Use Existing Remote Task]
Create New Remote Task

Existing Remote Task : rtBooks

The wizard prompts you for a remote task, or it lets you create a new one. For this tutorial you can use the remote task supplied in the tutorial library.

- In the wizard, select **rtBooks** from the Existing Remote Tasks list, and click on the Next button


OMNIS Remote Form Wizard - SideBar

Select and define the controls you wish to include on your new remote form class.

Control Type	Dataname

Control type : Single line entry field

Properties :

Dataname	Var1
Label	Label
Passwordchar	

The wizard now lets you enter the details for the fields you require on your form. You can specify the control type, dataname, and text label for each field on your form.

For the first field

- Select **Heading List** as the control type
- Enter **iBookList** as the dataname
- Enter **3** as the number of columns
- Enter **Title,Author,Cost** (no spaces) for the Column titles
- Enter **200,120,50** as the column widths
- When the above details are correct press the **Add** button

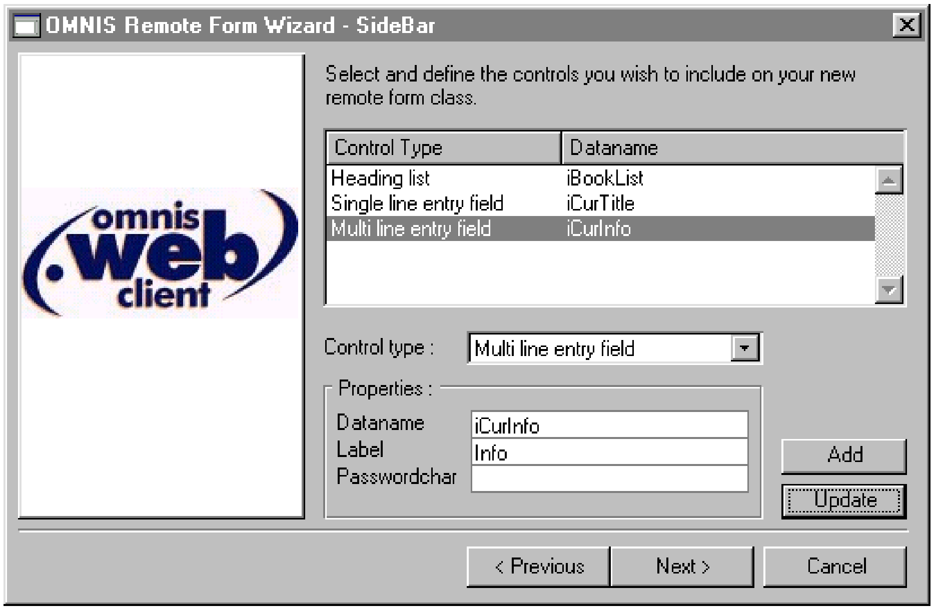
For the second field

- Select **Single line entry field** as the control type
- Enter **iCurTitle** as the dataname
- Enter **Title** as the label
- When the details are correct press the **Add** button

For the third field

- Select **Multi line entry field** as the control type
- Enter **iCurInfo** as the dataname
- Enter **Info** as the label
- Press the **Add** button when the details are complete

The wizard window should look like this:


OMNIS Remote Form Wizard - SideBar

Select and define the controls you wish to include on your new remote form class.

Control Type	Dataname
Heading list	iBookList
Single line entry field	iCurTitle
Multi line entry field	iCurInfo

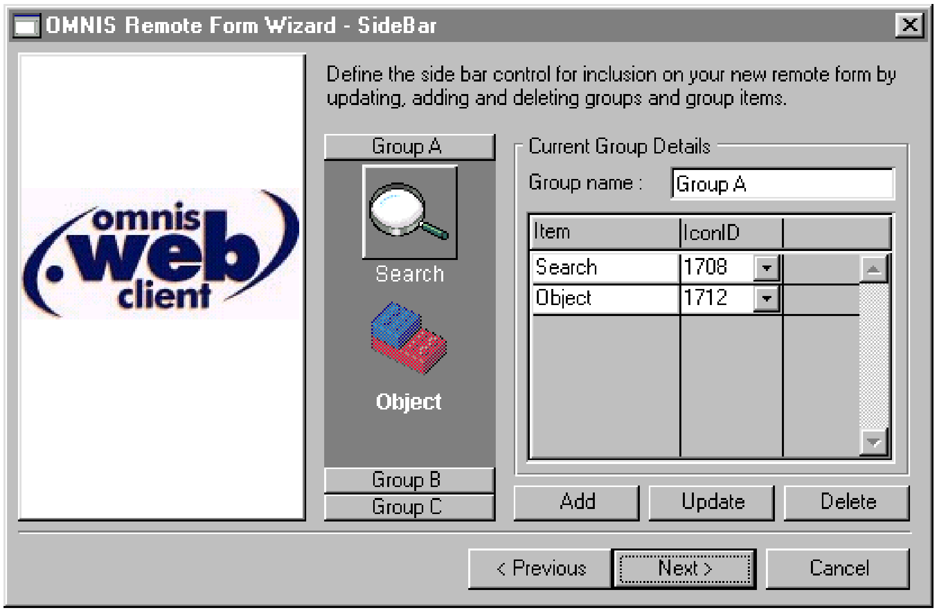
Control type : Multi line entry field

Properties :

Dataname	iCurInfo
Label	Info
Passwordchar	

When you are sure these details are correct

- Click on the **Next** button in the wizard window


OMNIS Remote Form Wizard - SideBar

Define the side bar control for inclusion on your new remote form by updating, adding and deleting groups and group items.

Current Group Details

Group name : Group A

Item	IconID
Search	1708
Object	1712

The wizard now lets you define the group and group items for the sidebar on your remote form. For our example remote form you need to create a sidebar with 5 groups. The wizard has three groups which you can update, and you need to add two further groups.

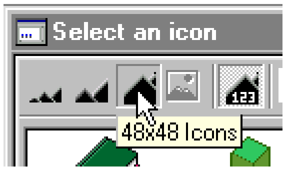
- In the Group name entry field replace the text Group A with **Bestsellers** and tab to the first item name in the list
- Replace the text with **Bestsellers** and tab to the Icon id cell

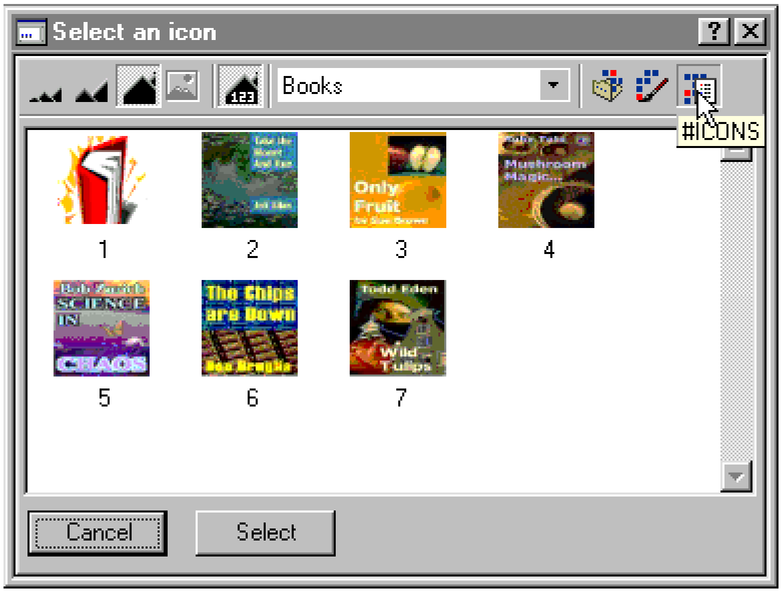
The icons for the Book browser sidebar are stored in the tutorial library in the #ICONS table, a system table stored in the library. You need to enter the Icon id of the icon for the Bestsellers group of books. To do this:

- Replace the current number in the IconID field with **1**

Alternatively, you can select the icon from the Select an Icon dialog.
To do this:

- In the wizard window, click on the dropdown arrow in the IconID field to open the Select an Icon dialog
- Click on the 48x48 Icons button in the toolbar in the dialog
- Next, click on the top right-most button marked #ICONS





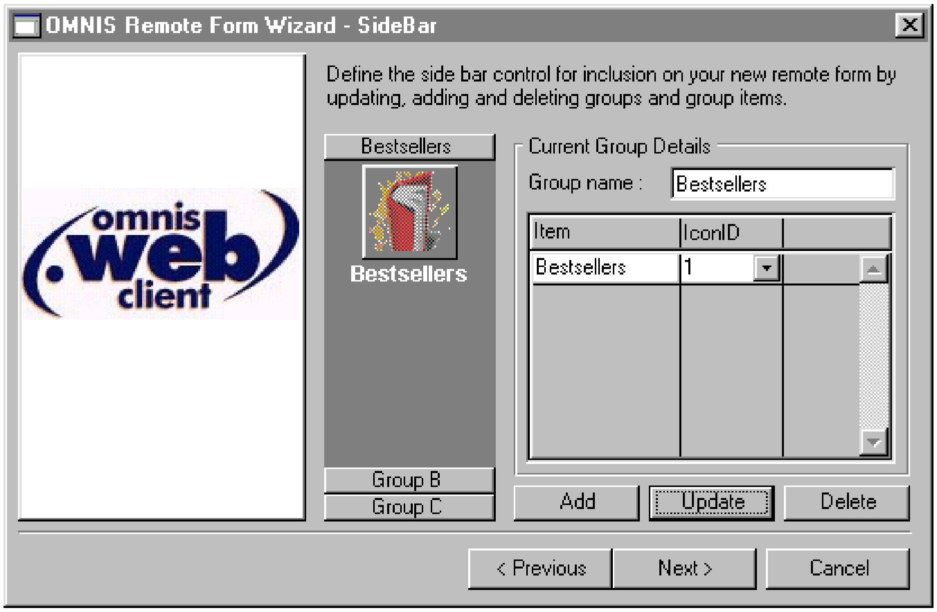
- In the Select Icon dialog, click on the icon numbered **1** and press the **Select** button

Back in the wizard, you don't need the second item in the group so you can delete it using the context menu over the item.

- Right-button click (Windows) or Ctrl-click (Mac) on the second line in the list and select the Delete option

- Click on the **Update** button to update the details for the first group

The wizard window should look like this:



Define the side bar control for inclusion on your new remote form by updating, adding and deleting groups and group items.

Group name : Bestsellers

Item	IconID
Bestsellers	1

- In the preview sidebar click on the Group B bar

- In the Group name entry field replace the text Group B with **Art & Design** and tab to the first item name in the list

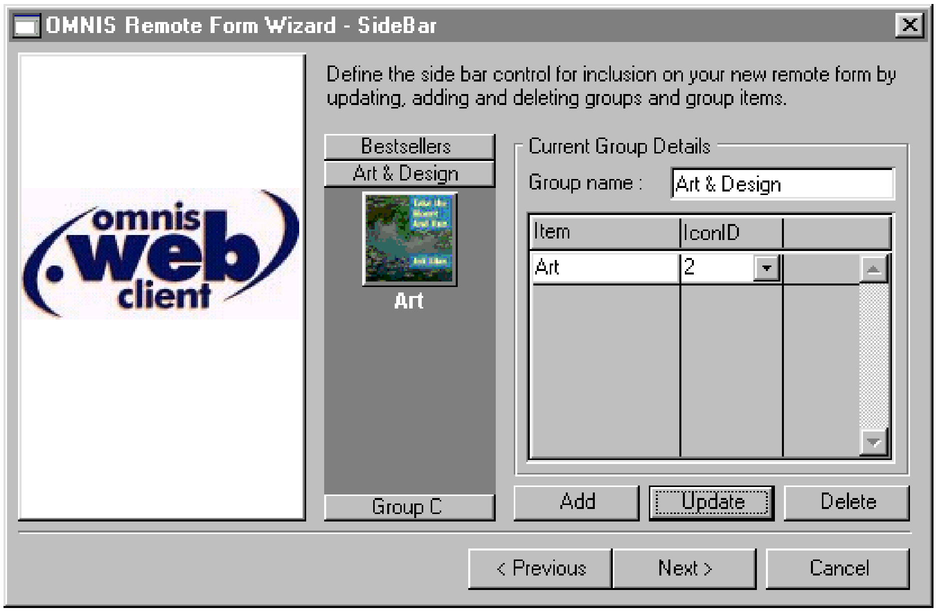
- Replace the text with **Art** and tab to the Icon id cell

- Replace the current number with **2** and tab to the second line in the list (or select the icon from the Select an Icon dialog as above)

- Right-button click (Windows) or Ctrl-click (Mac) on the second line in the list and select the Delete option

- Click on the **Update** button to update the details for the second group

Now the wizard should look like this:


OMNIS Remote Form Wizard - SideBar

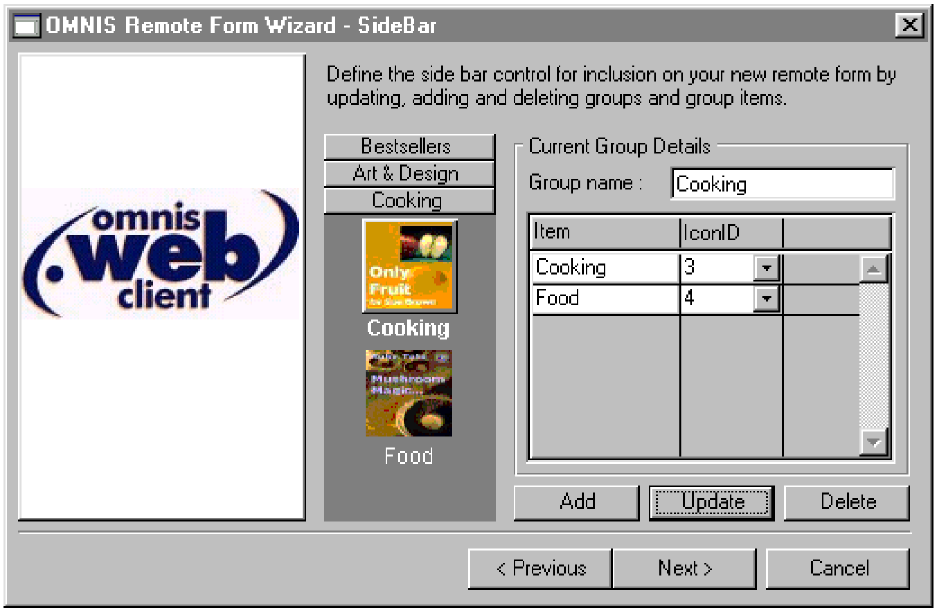
Define the side bar control for inclusion on your new remote form by updating, adding and deleting groups and group items.

Current Group Details

Group name : Art & Design

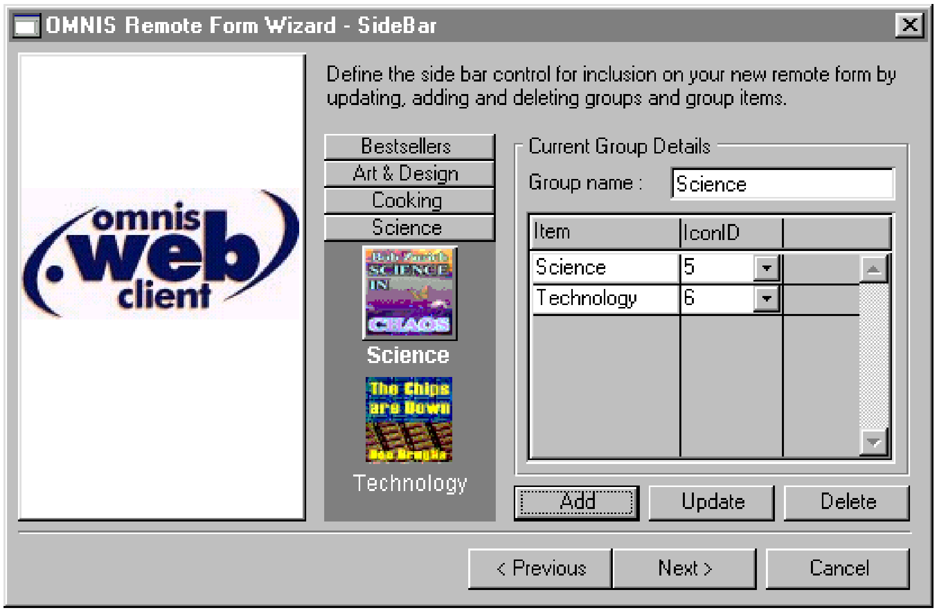
Item	IconID
Art	2

- In the preview sidebar click on the Group C bar
- In the Group name entry field replace the text Group C with **Cooking** and tab to the first item name in the list
- Replace the text with **Cooking** and tab to the Icon id cell
- Replace the current number with **3**
- Tab to the second line in the list to edit it
- Replace the current text with **Food** and tab to the Icon id cell
- Replace the current number with **4**
- Click on the **Update** button to update the details for the third group


OMNIS Remote Form Wizard - SideBar
Define the side bar control for inclusion on your new remote form by updating, adding and deleting groups and group items.
Bestsellers
Art & Design
Cooking
Cooking
Food
Current Group Details
Group name : Cooking
Add
Update
Delete
< Previous
Next >
Cancel

Item	IconID
Cooking	3
Food	4

- In the Group name entry field replace the text Cooking with **Science** and tab to the first item name in the list
- Replace the text with **Science** and tab to the Icon id cell
- Replace the current number with **5** and tab to the second line in the list
- Replace the text with **Technology** and tab to the Icon id cell
- Replace the current number with **6**
- This time rather than updating, you need to click on the **Add** button to add the details for the fourth group



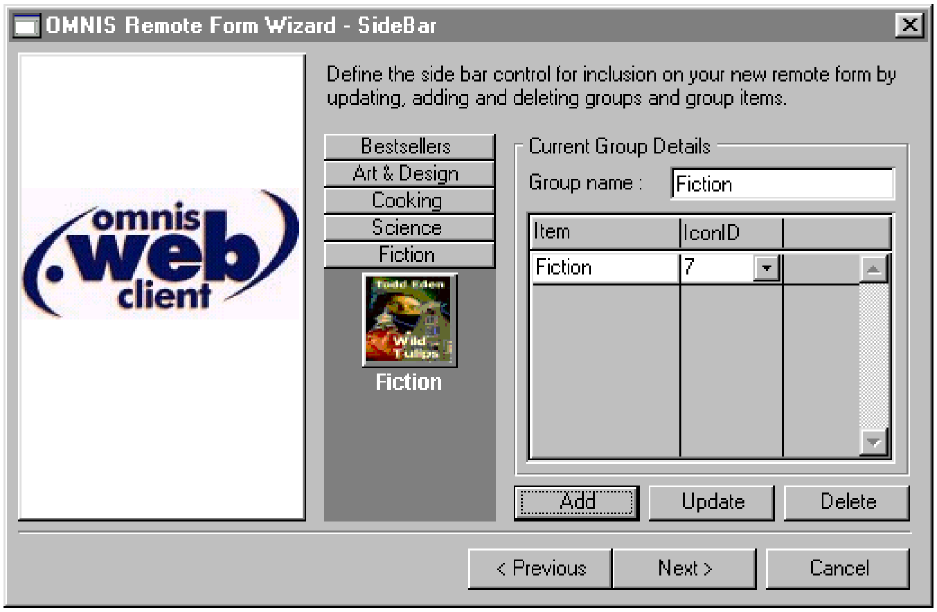
Define the side bar control for inclusion on your new remote form by updating, adding and deleting groups and group items.

Current Group Details

Group name : Science

Item	IconID
Science	5
Technology	6

- To enter the fifth group, in the Group name entry field replace the text Science with **Fiction** and tab to the first item name in the list
- Replace the text with **Fiction** and tab to the Icon id cell
- Replace the current number with **7** and tab to the second line in the list
- Right-button click (Windows) or Ctrl-click (Mac) on the second line in the list and select the Delete option
- Again rather than updating, you need to click on the **Add** button to add the details for the fifth group


OMNIS Remote Form Wizard - SideBar

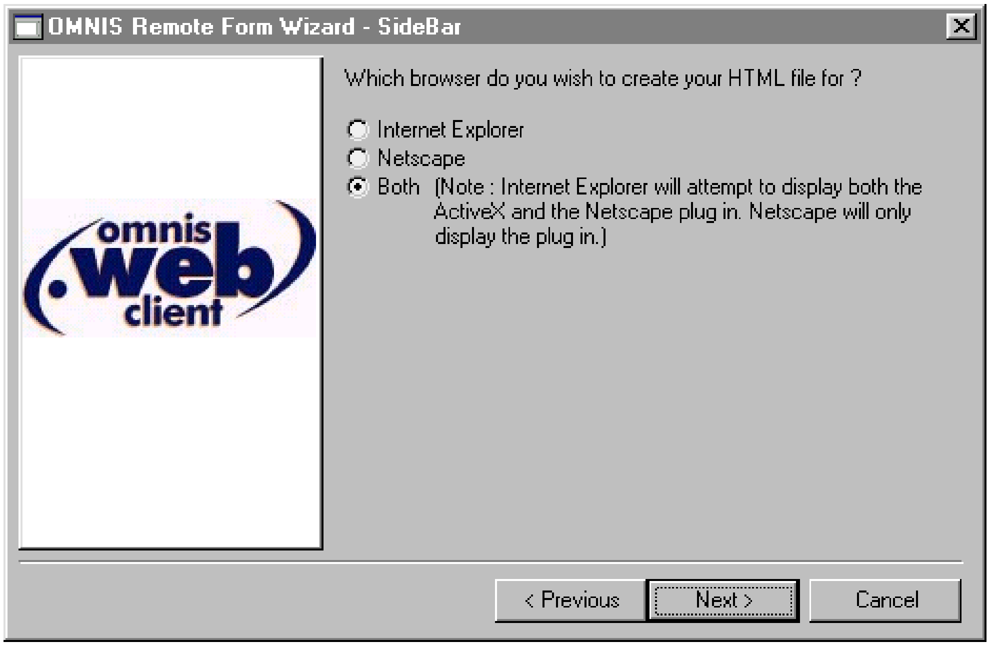
Define the side bar control for inclusion on your new remote form by updating, adding and deleting groups and group items.

Current Group Details

Group name : Fiction

Item	IconID
Fiction	7

- When all the group details, item names and icon ids for each group item are correct, click on the **Next** button


OMNIS Remote Form Wizard - SideBar

Which browser do you wish to create your HTML file for ?

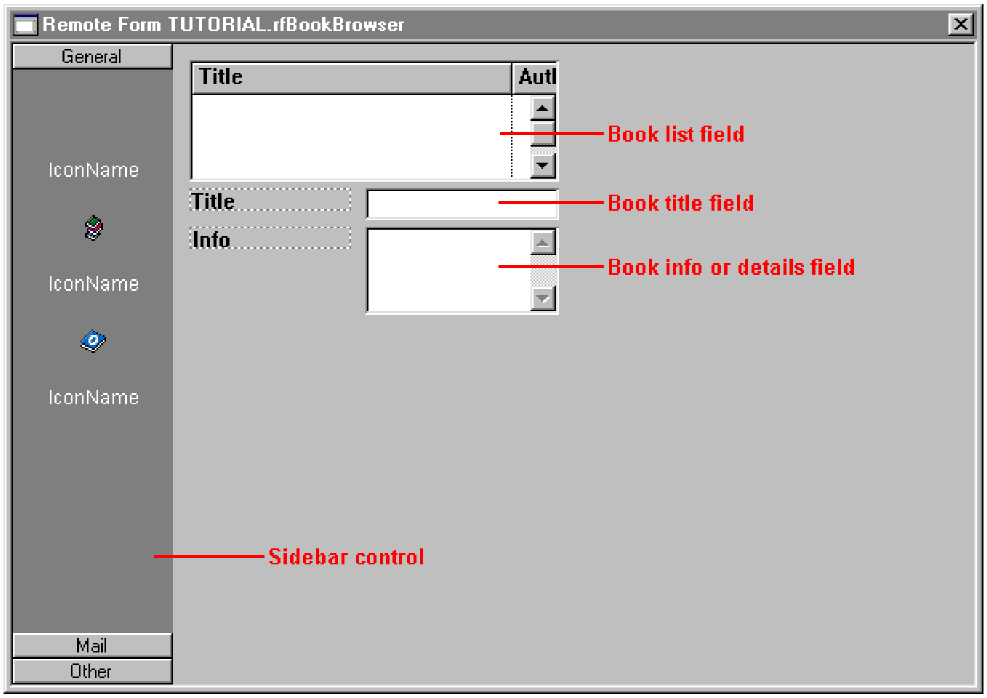
- Internet Explorer
- Netscape
- [✓ Both] (Note : Internet Explorer will attempt to display both the ActiveX and the Netscape plug in. Netscape will only display the plug in.)

The wizard now prompts you to choose the type of web browser you intend to use; this also depends on the type of OMNIS Web Client you installed before starting the tutorial.

- Select **Internet Explorer** or **Netscape** and click on the **Next** button

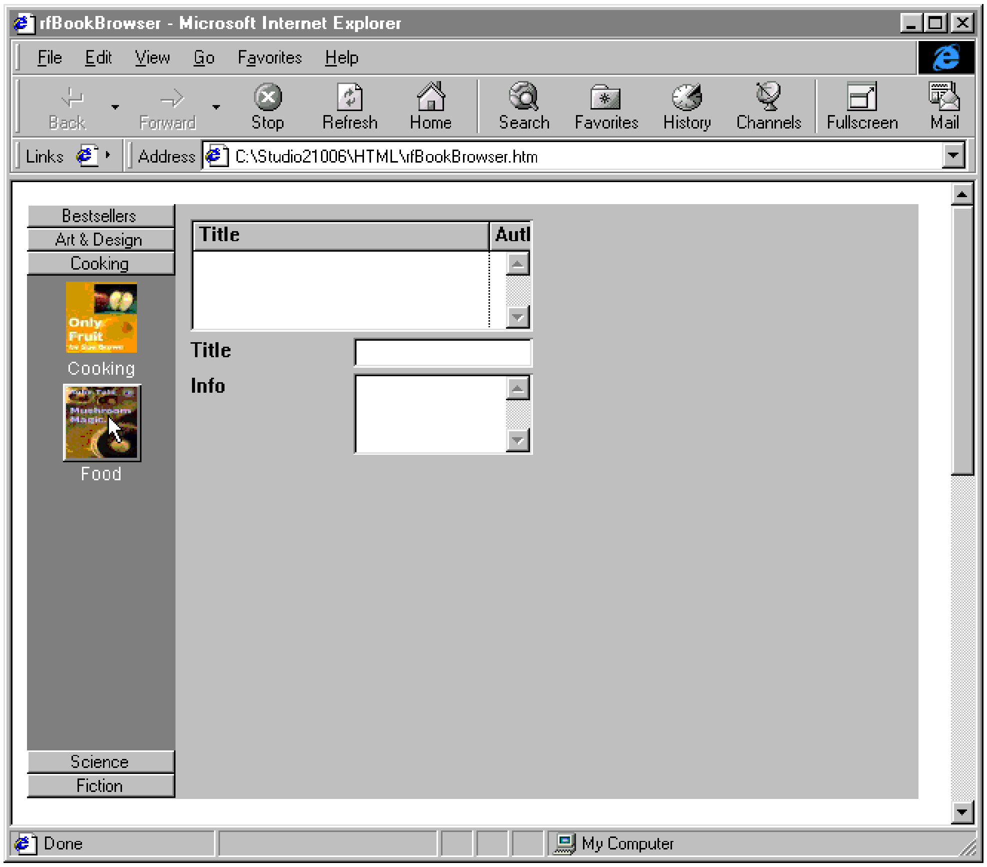
The **rfBookBrowser** remote form has been added to the tutorial library and should appear in the Class Browser. To view the form in design mode:

- Double-click on the **rfBookBrowser** remote form in the Class Browser



The form created by the Sidebar wizard contains the Sidebar control down the left-hand side and the three fields you specified, plus their text labels. The **BookList** (Heading List control type) is at the top which will hold the list of books, the **Title** field (single-line edit) is next which will display the current book title, and the **Info** field (multi-line edit) is at the bottom which will hold details about the currently selected book. Note that the Sidebar control does not show the groups you entered, they will however appear when you view the form in your browser.

This form does not contain any programming to fetch and display the Books data, but you can open or "test" the form at any time by pressing Ctrl/Cmnd-T. OMNIS opens your web browser automatically and displays the form.


rfBookBrowser - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail
Links Address C:\Studio21006\HTML\rfBookBrowser.htm
Bestsellers
Art & Design
Cooking
Cooking
Food
Science
Fiction
Title Autl
Title
Info
Done
My Computer

At this stage your form does not handle the Books data, but this gives you an idea of how the form appears in a web browser. You can however click on the Sidebar and change book groups. Note that the browser uses a template html page created for you by the form wizard. The template page is placed in the **html** folder in the main OMNIS folder. This html page has the OMNIS Web Client embedded in it by default with all the correct parameters set up for you automatically.

If your web browser containing the remote form does not open at this point, you should check that you have installed the OMNIS Web Client; if you installed the ActiveX check that it is registered properly. See the Troubleshooting guide for details about registering the OCX manually.

- Close or minimize the web browser to return to OMNIS; the remote form class should still be on top in OMNIS in design mode

Editing Form Fields

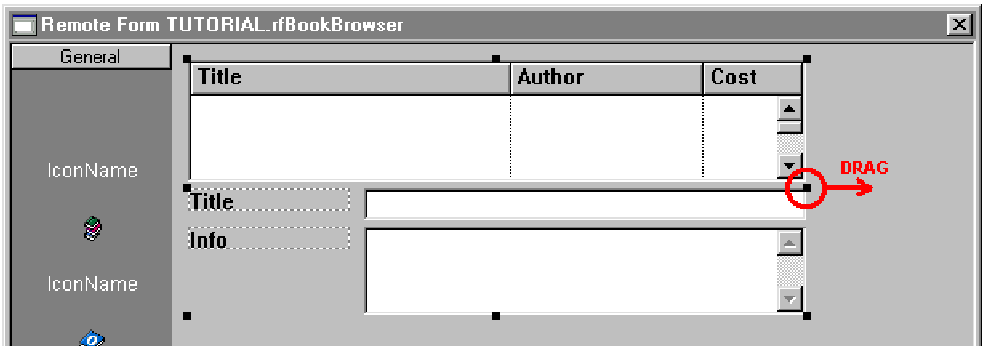
The next few steps involve resizing and renaming the three fields on the form, and adding some programming behind the remote form and, in particular, the BookList and Sidebar controls. Later you can add a picture field to show the book covers.

First, to resize the Book list, book title and info fields you can either resize each one in turn or select all three and resize them as a group.

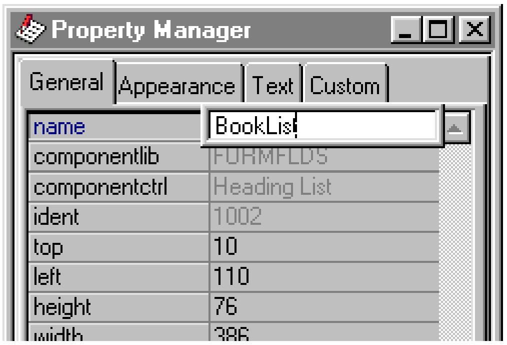
- In the remote form class, click on each field and resize it; the **Cost** column should be visible in the Booklist and you should resize the title and info fields to match

Alternatively, to resize them as a group:

- Shift-click on the three fields to select them all
- Drag the right handle of the group selection to make all three fields wider, as shown below



- Click on the background of the form to deselect all the fields
- At this stage you may like to make the Info field deeper by dragging its bottom handle
- Next, click on the BookList at the top and press F6/Cmnd-6 to open the Property Manager, or bring it to the top
- On the General tab in the Property Manager change the **name** property to **BookList**



name	BookList
componentlib	FORMFLDS
componentctrl	Heading List
ident	1002
top	10
left	110
height	76
width	386

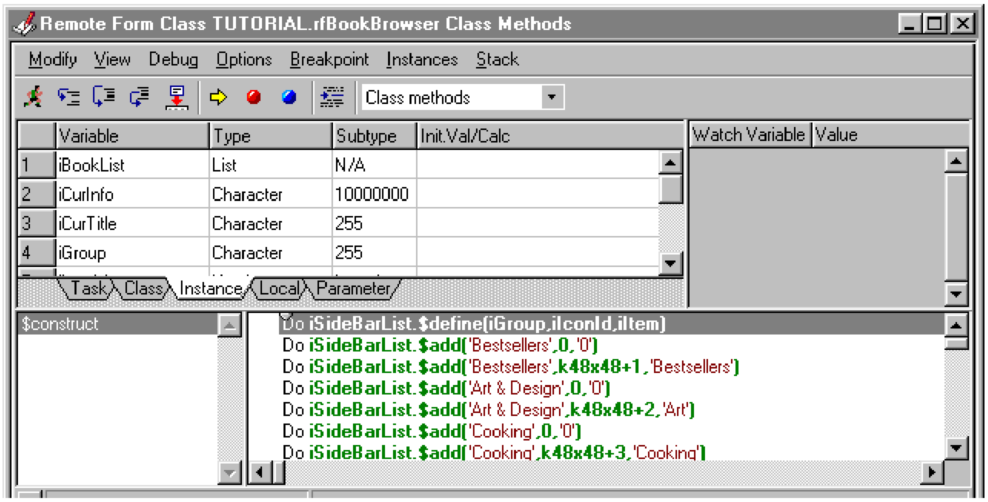
- Similarly, click on the Title field (the single line field), open the Property Manager, and change its name to **Title**

- Then click on the Info field (the multi-line one), open the Property Manager, and change its name to **Info**

Adding Variables and Methods to the Form

Having altered the fields you must add some programming behind the form and its controls. You need to add some variables to the form, edit some of the code that the wizard created for you, and add one or two new methods.

- Double-click on the background of the form, in design mode, to open the method editor for the remote form class



	Variable	Type	Subtype	Init.Val/Calc
1	iBookList	List	N/A	
2	iCurInfo	Character	10000000	
3	iCurTitle	Character	255	
4	iGroup	Character	255	

```
Do iSideBarList.$define(iGroup,iIconId,iItem)
Do iSideBarList.$add('Bestsellers',0,'0')
Do iSideBarList.$add('Bestsellers',k48x48+1,'Bestsellers')
Do iSideBarList.$add('Art & Design',0,'0')
Do iSideBarList.$add('Art & Design',k48x48+2,'Art')
Do iSideBarList.$add('Cooking',0,'0')
Do iSideBarList.$add('Cooking',k48x48+3,'Cooking')
```

- Click on the **Instance** tab in the variable pane

- Right-click/Ctrl-click on the variable pane and select **Insert New Variable** from the context menu
- Name the variable **iBookGroup** and select Type **Number** and Subtype **Long Integer** from the appropriate droplists
- To insert another variable, Right-click/Ctrl-click on the variable pane and select **Insert New Variable** from the context menu
- Name the variable **iCurCover** and select **Picture** from the Type droplist (it doesn’t require a subtype)

You should now have 9 variables in your remote form. Next you are going to edit the code in the $construct method, which should be selected in the method editor. This method builds the group items and icon details for the Sidebar control in the form. First, edit the first line of code:

```
Do iSideBarList.$define(iGroup,iIconId,iItem)
```

by adding the **iBookGroup** variable to the list definition. The line should now be:

```
Do iSideBarList.$define(iGroup,iIconId,iItem,iBookGroup)
```

Next you need to add the book groups or categories to each line of code that defines each one of the group items in the Sidebar control. The following table gives you the categories (the same as stored in the Books data file).

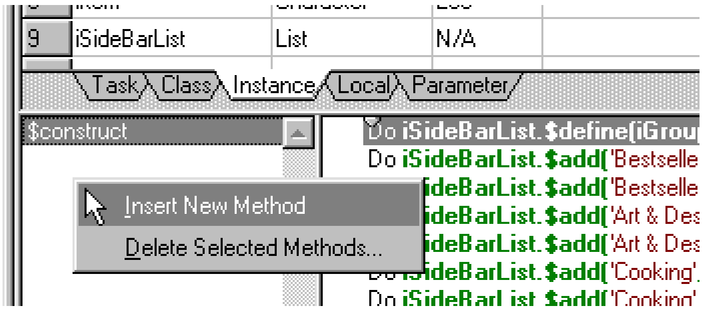
Bestsellers	0
Art	1
Cooking	2
Food	3
Science	4
Technology	5
Fiction	6

The lines you need to edit are shown highlighted (the additions are underlined and in red), and shows you the final method. If you prefer, you can Copy the method from the on-line PDF manual and Paste it into the method editor replacing the existing code, or you can copy the complete method from the rfBookBrowserFinal class in the tutorial library.

```
Do iSideBarList.$define(iGroup,iIconId,iItem,iBookGroup)
Do iSideBarList.$add('Bestsellers',0,'0')
Do iSideBarList.$add('Bestsellers',k48x48+1,'Bestsellers',0)
Do iSideBarList.$add('Art & Design',0,'0')
Do iSideBarList.$add('Art & Design',k48x48+2,'Art',1)
Do iSideBarList.$add('Cooking',0,'0')
Do iSideBarList.$add('Cooking',k48x48+3,'Cooking',2)
Do iSideBarList.$add('Cooking',k48x48+4,'Food',3)
Do iSideBarList.$add('Science',0,'0')
Do iSideBarList.$add('Science',k48x48+5,'Science',4)
Do iSideBarList.$add('Science',k48x48+6,'Technology',5)
Do iSideBarList.$add('Fiction',0,'0')
Do iSideBarList.$add('Fiction',k48x48+7,'Fiction',6)
```

You can save the form at any time by Selecting the **File>>Save rfBookBrowser** from the main OMNIS menubar. Next you are going to add a method to the remote form.

- In the method editor, Right-click/Ctrl-click on the method list just below the $construct method, and select **Insert New Method** from the context menu

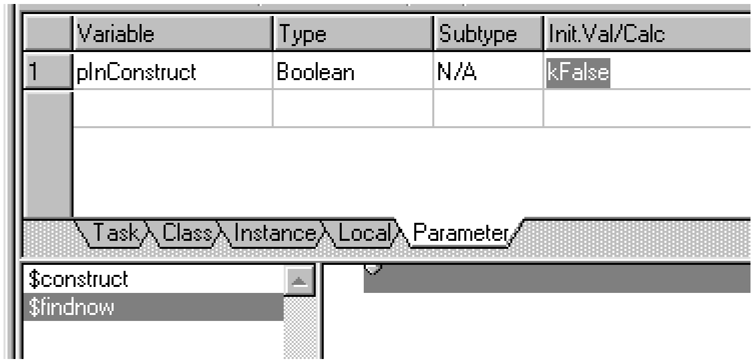


```
Do iSideBarList.$define(iGrou
Do iSideBarList.$add('Bestselle
...ideBarList.$add('Bestselle
...ideBarList.$add('Art & Des
...ideBarList.$add('Art & Des
...ideBarList.$add('Cooking',
```

- Name the new method **$findnow**

- Click on the **Parameter** tab in the variable pane and add a new variable by Right-clicking/Ctrl-clicking

- Name the new variable **pInConstruct**, select **Boolean** from the Type droplist, and give it an Initial value of **kFalse**



	Variable	Type	Subtype	Init.Val/Calc
1	pInConstruct	Boolean	N/A	kFalse

$construct
$findnow

- Click in the method editor again and enter the following code for the **$findnow** method

Again, to save typing in this method you can copy the code from the on-line PDF manual and paste it into the $findnow method, or you can copy and paste the complete method from the rfBookBrowserFinal class in the tutorial library. Some lines may be commented out, but try un-commenting these lines individually.

```
Set main file {fBooks}
Set current list iBookList
Load from list
Find on fBooks.BookTitle (Exact match)
Calculate iCurCover as fBooks.BookCover
Calculate iCurInfo as fBooks.BookInfo
Calculate iCurTitle as fBooks.BookTitle

Do $cinst.$objs.Info.$redraw()
Do $cinst.$objs.Cover.$redraw()
Do $cinst.$objs.Title.$redraw()

If not(pInConstruct)
   Do $cinst.$senddata(iCurCover,iCurInfo,iCurTitle)
End If
```

Next you need to add some code to the $construct() method to call the $findnow() method.

- Click on the **$construct** method and scroll to the bottom of the lines of code in the method

- Add the following lines of code; if you prefer, try Copying and Pasting the code as before

```
;  Build book list
Calculate iBookList as tBookList
Set current list iBookList
Calculate #L as 1
Do method $findnow (kTrue)
```

The complete $construct method, including all the edits and additions, is shown below; if you have difficulty with this method, you can copy the complete method from the rfBookBrowserFinal class in the tutorial library.

```
Do iSideBarList.$define(iGroup,iIconId,iItem,iBookGroup)
Do iSideBarList.$add('Bestsellers',0,'0')
Do iSideBarList.$add('Bestsellers',k48x48+1,'Bestsellers',0)
Do iSideBarList.$add('Art & Design',0,'0')
Do iSideBarList.$add('Art & Design',k48x48+2,'Art',1)
Do iSideBarList.$add('Cooking',0,'0')
Do iSideBarList.$add('Cooking',k48x48+3,'Cooking',2)
Do iSideBarList.$add('Cooking',k48x48+4,'Food',3)
Do iSideBarList.$add('Science',0,'0')
Do iSideBarList.$add('Science',k48x48+5,'Science',4)
Do iSideBarList.$add('Science',k48x48+6,'Technology',5)
Do iSideBarList.$add('Fiction',0,'0')
Do iSideBarList.$add('Fiction',k48x48+7,'Fiction',6)

;  Build book list
Calculate iBookList as tBookList
Set current list iBookList
Calculate #L as 1
Do method $findnow (kTrue)
```

- When the $construct method is complete, close the method editor

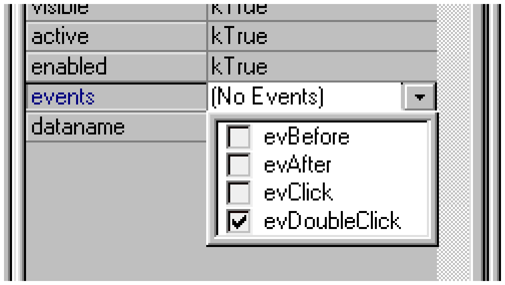
At this stage you can save the changes you’ve made to the form by selecting the **File>>Save rfBookBrowser** from the main OMNIS menubar. You still have to add some code behind the BookList and Sidebar controls, but you may like to try the form in your web browser again. When you open the form this time, using Ctrl/Cmnd-T, the BookList should contain the books from the Bestsellers category, but at this stage clicks on the Sidebar and the BookList won’t have any effect.

- To return to OMNIS, close or minimize your web browser

Events and Adding Methods to Form Fields

Next you are going to add some code behind the BookList. First though you need to enable the evDoubleClick event for the list. Normally, all field events are enabled for window objects, but for remote form objects you need to explicitly enable which events you want to be processed by the object. You can do this by setting the object's **events** property.

- In the remote form, click on the **BookList** field and press F6/Cmnd-6 to open the Property Manager or bring it to the top
- Under the General tab, click on the **events** property and select/check the **evDoubleClick** event in the droplist



visible	kTrue
active	kTrue
enabled	kTrue
events	(No Events)
dataname	

☐ evBefore
☐ evAfter
☐ evClick
☑ evDoubleClick

Now you need to add the method for the BookList.

- In the remote form, double-click on the **BookList** field to open the method editor for the object
- Enter the following code in the $event method

```
On evDoubleClick
   Do method $findnow
```

- When you've entered the method, close the method editor window

Next you are going to enter the $event() method for the Sidebar control. This time however, you do not need to enable any events for the Sidebar control, that is evIconPicked, since the Sidebar wizard has done this for you automatically. To enter the event handler for the Sidebar

- Double-click on the **Sidebar** control in the form

- In the method editor, enter the following code in the $event method

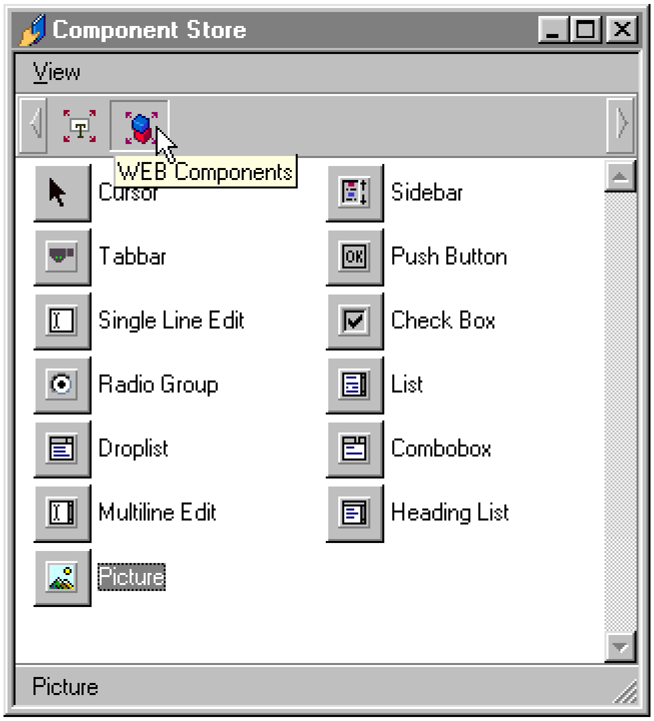
```
On evIconPicked
   Set current list iSideBarList
   Load from list {pLinenum}
   Do method $ctask.$buildbooks (iBookGroup)
   Calculate iBookList as tBookList
   Do $cinst.$objs.BookList.$redraw()
   Do $cinst.$senddata(iBookList)
```

- When you've entered the method, close the method editor window, and save the class using **File>>Save rfBookBrowser**

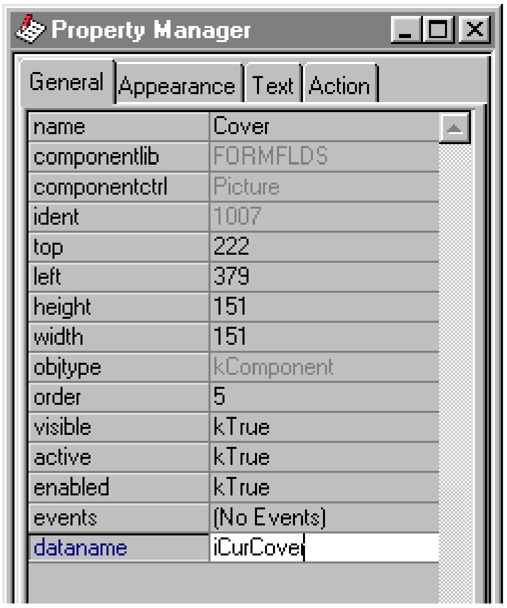
Adding a Picture Field

And finally you are going to add a picture field to the form to view the book covers. You can create picture fields in the Sidebar wizard, but this section lets you add a field to your remote form using the Component Store.

- Assuming the form is on top, press F3/Cmnd-3 to open the Component Store, and click on the **WEB Components** button in the Component Store toolbar



- Drag the **Picture** object onto your form and release the mouse
- With the Picture selected, press F6/Cmnd-6 to open the Property Manager, or bring it to the top
- Under the General tab, change the **name** property to **Cover**
- Enter the **dataname** as **iCurCover**



name	Cover
componentlib	FORMFLDS
componentctrl	Picture
ident	1007
top	222
left	379
height	151
width	151
objtype	kComponent
order	5
visible	kTrue
active	kTrue
enabled	kTrue
events	(No Events)
dataname	iCurCover

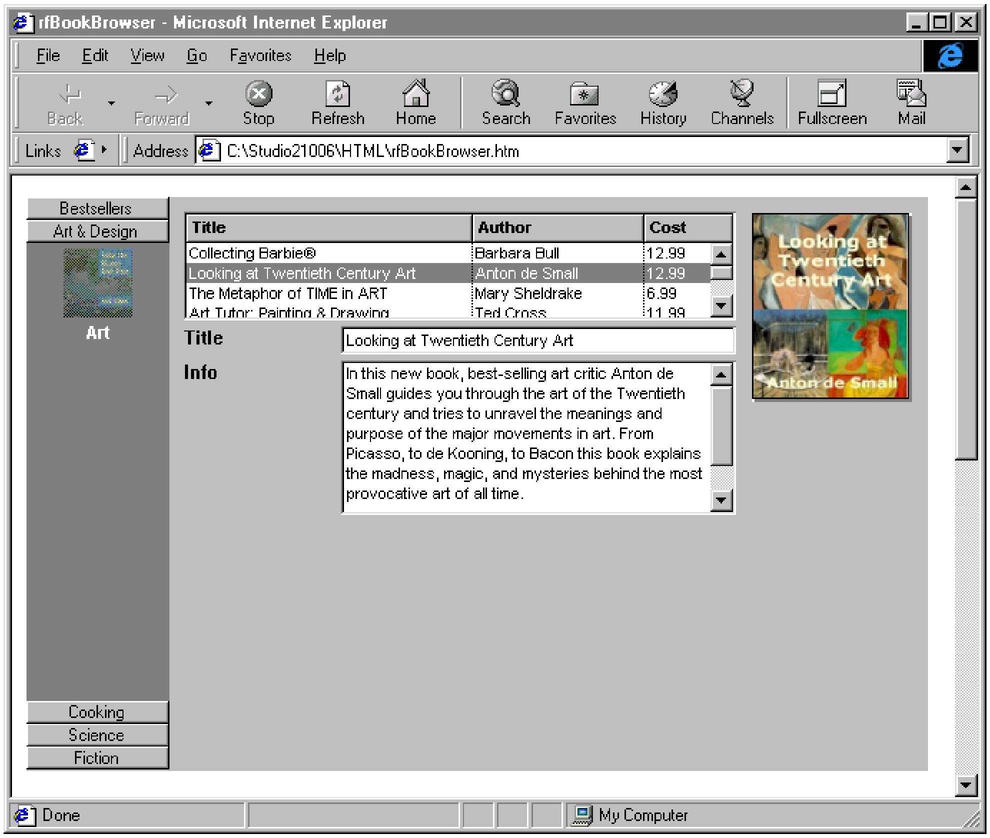
You may like to remove the scroll bars from the picture field, and resize and position it in the form. To do this:

- Click on the Picture field and press F6/Cmnd-6 to open the Property Manager, or bring it to the top
- Under the Appearance tab, set both the properties **vertscroll** and **horzscroll** to kFalse
- Under the Appearance tab, set the **effect** property to **kShadowBorder**
- Under the General tab, set the **height** property to **132,** and the **width** property to **112**
- Make the form a bit wider and move the picture field up to the top right-hand corner of the form, to the right of the BookList field

Testing the Form and Browsing the Books

To test the remote form:

- Press Ctrl/Cmnd-T to try out the form, and browse the Books data



rfBookBrowser - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail
Links Address C:\Studio21006\HTML\rfBookBrowser.htm
Bestsellers
Art & Design
Art
Cooking
Science
Fiction
Done
My Computer

Title	Author	Cost
Collecting Barbie®	Barbara Bull	12.99
Looking at Twentieth Century Art	Anton de Small	12.99
The Metaphor of TIME in ART	Mary Sheldrake	6.99
Art Tutor: Painting & Drawing	Ted Cross	11.99

Title: Looking at Twentieth Century Art

Info: In this new book, best-selling art critic Anton de Small guides you through the art of the Twentieth century and tries to unravel the meanings and purpose of the major movements in art. From Picasso, to de Kooning, to Bacon this book explains the madness, magic, and mysteries behind the most provocative art of all time.

This form allows you to browse the book details. You can click on the Sidebar to change the group or subgroup to view the different categories of books, and you can ***double-click*** on the BookList to view the details and cover for individual books.

Debugging the Form

When you are developing your remote form you can set breakpoints in your code in the method editor, and use trace mode. If you set a breakpoint in one of the methods, open the remote form in your web browser (using Ctrl/Cmnd-T) and use the form, OMNIS will switch back to the debugger in design mode. Similarly if trace mode is enabled OMNIS will trace all the methods that are called in the form, allowing you to see what's going on.

Deploying the Remote Form

To deploy an OMNIS web solution using the OMNIS Web Client you need a Win32 web server, and a Win32 server to install the runtime version of the OMNIS executable. However, for the purposes of this tutorial you can use a Personal web server set up in Microsoft FrontPage and the development version of OMNIS on your local machine. The deployment process is exactly the same in both cases, which involves:

– installing and configuring the OMNIS server, that is, the OMNIS runtime and your OMNIS library

– installing and configuring your web server, and editing your html pages

If you go ahead and deploy the tutorial library to your web server, rather than just using it locally, you must edit the $construct() method in the Startup_Task. In particular you need to edit the Open data file command and associated code that opens the Webclient datafile, and you must remove the *Prompt for data file* command. You must not use any commands in the OMNIS server application that prompt the user in any way.

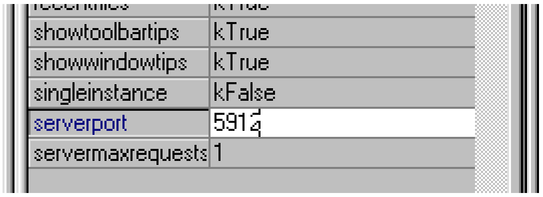
OMNIS Server

The tutorial uses the development version of OMNIS, but if you are deploying the library to the web you must install and serialize the OMNIS runtime.

For the OMNIS development version:

- Start OMNIS, open the Tutorial library and connect to the Books data file

- Select Tools>>Options from the main OMNIS menubar



showtoolbartips	kTrue
showwindowtips	kTrue
singleinstance	kFalse
serverport	5912
servermaxrequests	1

- In the OMNIS preferences in the Property Manager, set the **serverport** property to 5912 or if this number is already in use choose some other number

- Restart OMNIS, reopen the library and keep it running in the background

For the OMNIS runtime:

- Install the OMNIS runtime onto your Win32 server and serialize it using a web serial number

- Copy the tutorial library and the datafile containing the Bookstore data (Webclient.df1) to your Win32 server
- Start the OMNIS runtime
- Select **File>>Server Configuration** from the main OMNIS menubar
- Enter 5912 in the port number field and close the dialog
- Open the Tutorial library and connect to the Books data file; keep the library running in the background

Web Server and HTML files

The tutorial uses a Personal web server set up using MS FrontPage, but if you are deploying the application to the web you must purchase and install a standard Win32 web server. You may be able to get a free or trial version of a web server from the Internet, or purchase a basic web server suitable for testing purposes. In either case the web server configuration is the same.

- Install your web server, or set up a personal web server in MS FrontPage; see the Help in MS FrontPage for details about creating a new FrontPage web
- Go to your OMNIS development tree and locate the OMNIS/WebClient/Server folder
- Copy the OMNISAPI.DLL file to your web server **cgi-bin** folder
- Next, locate the **Html** folder in the main OMNIS folder
- Copy the **rfBookBrowser.htm** file to your personal or local web server
- Open the **rfBookBrowser.htm** file in a text editor or Html source editor
- In the Html source for the rfBookBrowser.htm file, edit the following parameters, in particular you need to edit the WebServerURL and WebServerScript parameters

For Internet Explorer you need to edit the following parameters for the ActiveX:

```
<param name="OmnisServer" value="5912">  ;; the port number
<param name="OmnisLibrary" value="TUTORIAL">
   ; the library name less the file extension
<param name="OmnisClass" value="rfBookBrowser">
   ; the remote form name
<param name="WebServerURL" value="http://www.yourdomain.com">
   ; the domain name or IP address of your web server or
   ; personal web server
<param name="WebServerScript" value="/cgi-bin/omnisapi.dll">
   ; the location of the omnis web server extension
```

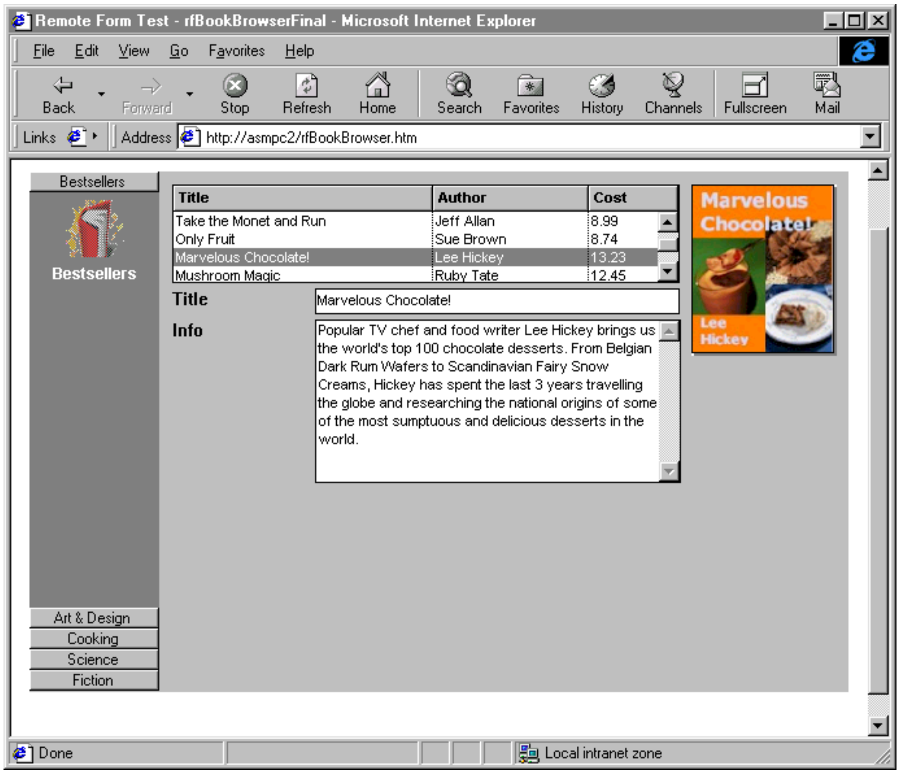
The parameters for the Netscape plug-in should be as follows:

```
<EMBED type=application/OMNIS-RCC-plugin name="rcc1"
   width=500 height=500
   OmnisServer="5912"  ;; the port number
   OmnisLibrary="TUTORIAL"
   ; the library name less the file extension
   OmnisClass="rfBookBrowser"
   ; the remote form name
   WebServerURL="http://www.yourdomain.com"  ; the domain name or IP
   address of your web server or personal web server
   WebServerScript="/cgi-bin/omnisapi.dll"  ; the location of the
   omnis web server extension
   Param1="pOption1=Data" .. .. Param9="pOption9=Data">
```

The port number, library name, and remote form name will already be set up for you, but if they’re not enter the values shown above. If your web server does not support the MS ISAPI interface you will have to use the nph-omniscgi.exe cgi program. In this case, you need to add the nph-omniscgi.exe program to your cgi-bin folder and use this name in the **WebServerScript** parameter in your HTML file(s).

- Save and close the rfBookBrowser.htm file
- If you are deploying to a remote web server, copy the rfBookBrowser.htm file to your web server using FTP
- Open your web browser and navigate to the rfBookBrowser.htm file using “[YourDomainName or IPAddress]/rfBookBrowser.htm”, where YourDomainName could be http://www.yourdomain.com, or IPAddress might be the IP address of your personal web server

The Book Browser form should appear and you can start to browse the books data.


Remote Form Test - rfBookBrowserFinal - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail
Links Address http://asmpc2/rfBookBrowser.htm
Bestsellers
Art & Design
Cooking
Science
Fiction
Done
Local intranet zone

Title	Author	Cost
Take the Monet and Run	Jeff Allan	8.99
Only Fruit	Sue Brown	8.74
Marvelous Chocolate!	Lee Hickey	13.23
Mushroom Magic	Ruby Tate	12.45

Title: Marvelous Chocolate!

Info: Popular TV chef and food writer Lee Hickey brings us the world's top 100 chocolate desserts. From Belgian Dark Rum Wafers to Scandinavian Fairy Snow Creams, Hickey has spent the last 3 years travelling the globe and researching the national origins of some of the most sumptuous and delicious desserts in the world.

Viewing the OMNIS Web Client Example Library

You can see an enhanced version of the Books form in the OMNIS Web Client example library available in the Examples Browser. The enhancements include a shopping basket to store the books you want to buy, and a checkout facility for entering credit card details. You can open the Bookstore library via the Examples Browser using the **Tools>>Examples Browser** option on the main OMNIS menu. You can also view the Bookstore application on the Internet via the OMNIS Software website www.omnis-software.com.

Chapter 3–Designing OMNIS Remote Forms

To create an OMNIS web solution you need to create a library containing your OMNIS remote form classes and remote task classes. You can create and test the remote forms on your own machine using your own web browser, that is, you can develop your OMNIS library before installing your web server, the web server extension and OMNIS Server, or before setting up your html pages; these are described later in this manual.

General information about developing OMNIS applications, designing forms and programming in OMNIS, are found in the *Using OMNIS Studio* and *OMNIS Programming* manuals available with OMNIS Studio.

You can create OMNIS remote forms from scratch or using the templates and wizards available in the Component Store. The wizards give you complete control over the fields and controls contained in your form and save you a lot of time. All the wizards create a new remote task or use an existing task to link to the new remote form.

Creating Remote Forms using Wizards

You can create a remote form using one of the wizards available in the Component Store. The wizards let you specify the fields and controls you want on your form. The wizards are summarized here and described in more detail below.

– **Plain**
  creates a blank form and links it to a remote task; you must add your own instance variables and form controls to this form

– **Tabs**
  creates a remote form containing a tab strip and page pane control and fields on each pane; creates an instance variable for each form field

– **Wizard**
  creates a remote form containing a page pane control, fields on each pane, and Next, Previous, and Finish buttons; creates an instance variable for each form field

– **Submit**
  creates a standard submit form containing your fields, Submit and Clear buttons; creates an instance variable for each form field

- **Password**
  creates a standard password form containing Password and Username fields; creates an instance variable for each form field
- **Sidebar**
  creates a form containing a number of fields and a sidebar control; creates an instance variable for each form field
- **Sidebar with pages**
  creates a form containing a page pane and a sidebar control; you specify the fields for each pane and link each pane to a particular group in the sidebar; creates an instance variable for each form field
- **Multiform**
  lets you create a complete website containing home page, navigation bar, and any number of remote forms created using any of the other wizards

All the wizards prompt you for a remote task class: a *remote task* instance manages the connection to the client, and is a container for all the instances associated with a particular remote form. You can select an existing remote task in your library, or you create one based on the templates provided. For most types of remote form you can select the Plain Remote Task template.

All wizards, except the Plain one, generate an html template which is placed in the **html** folder of the OMNIS root. This html page lets you test the form in design mode using Ctrl/Cmnd-T and provides a template for your final html pages for your website. During each wizard you are prompted to select the type of browser you want to create html pages for. You can select Internet Explorer or Netscape, or both, and in all cases the appropriate plug-in is embedded into your template html pages.

All wizards that create forms containing data fields or controls, prompt you for the **dataname** and control type of each field you require on your form. An instance variable is created for every field you add to a form.

All remote forms created using a wizard have a $construct() method containing a parameter variable called *pParams* of type *Row Variable*. This row variable contains the parameters in the remote form.

Having created your remote form(s) using the Remote Form Wizards, you can modify them or add instance variables and form controls, as appropriate, to complete your remote forms. The remainder of this chapter describes the different form wizards and form controls available in OMNIS.

Plain Form Wizard

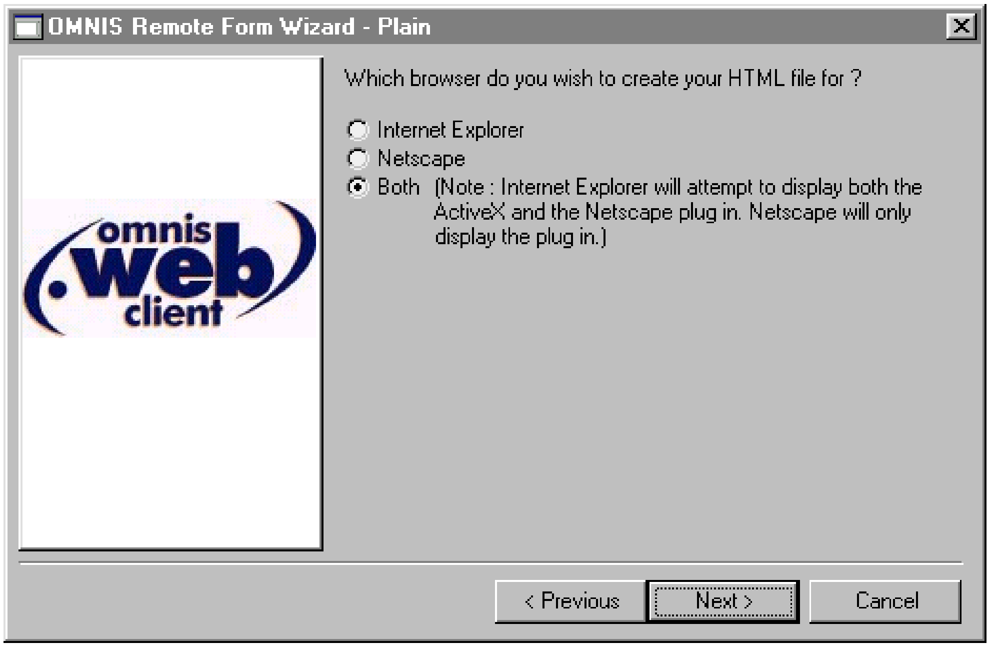
The Plain form wizard creates a blank form and links it to a remote task. The wizard prompts you for an existing remote task or lets you create a new one based on one of the templates provided. You need to add your own instance variables and form controls to this type of remote form.

To create a Plain Form

- Open your library and display its classes in the Class Browser
- Open the Component Store and click on the Remote Form Classes button
- Select the Plain remote form wizard and drag its icon onto your library in the OMNIS Class Browser
- Name the new remote form and press Return
- Select an existing remote task from the list of tasks provided, and click on the Next button

or

- Click on the Create New Task option, select a template from the list provided, name the new remote task, and click on the Next button



Which browser do you wish to create your HTML file for ?

Internet Explorer
Netscape
[✓ Both] (Note : Internet Explorer will attempt to display both the ActiveX and the Netscape plug in. Netscape will only display the plug in.)

- When you have selected your browser option, click on the Next button

OMNIS creates a new remote form in your library. Double-click on the form in the Class Browser to modify it.

Tabs Form Wizard

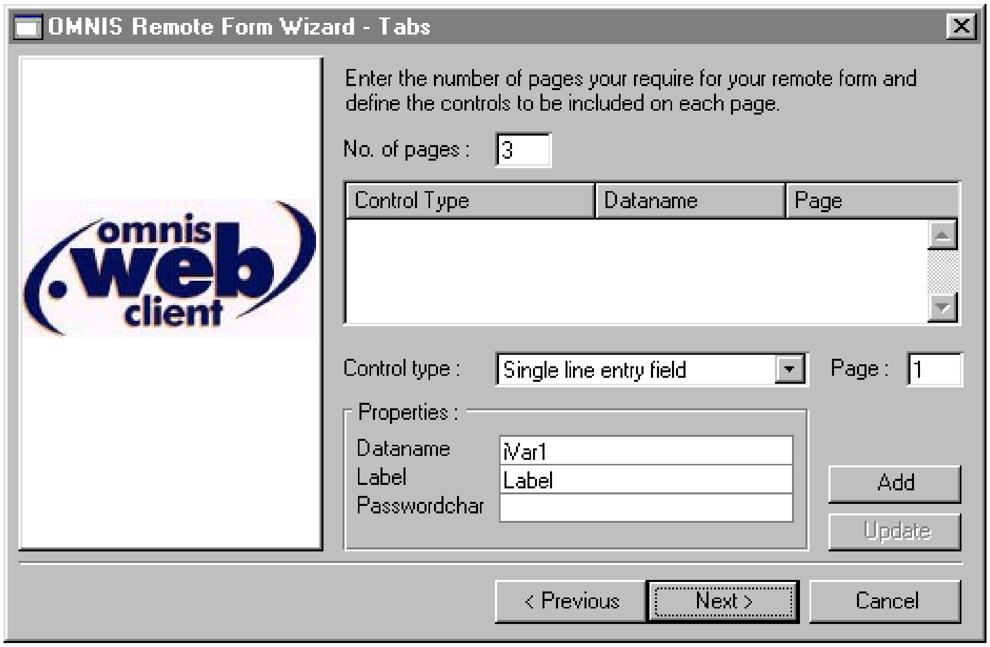
The Tabs form wizard creates a remote form containing a tab strip and page pane control. In the wizard you can add fields and controls to each pane in the form. The resulting form lets the user enter data on a number of panes by clicking the tabs to change page.

To create a Tabs Form

- Open your library and display its classes in the Class Browser
- Open the Component Store and click on the Remote Form Classes button
- Select the Tabs remote form wizard and drag its icon onto your open library in the OMNIS Class Browser
- Name the new remote form and press Return
- Select an existing remote task from the list of tasks provided, and click on the Next button

or

- Click on the Create New Task option, select a template from the list provided such as the Plain Remote Task, name the new remote task, and click on the **Next** button


OMNIS Remote Form Wizard - Tabs

Enter the number of pages your require for your remote form and define the controls to be included on each page.

No. of pages : 3

Control Type	Dataname	Page

Control type : Single line entry field

Page : 1

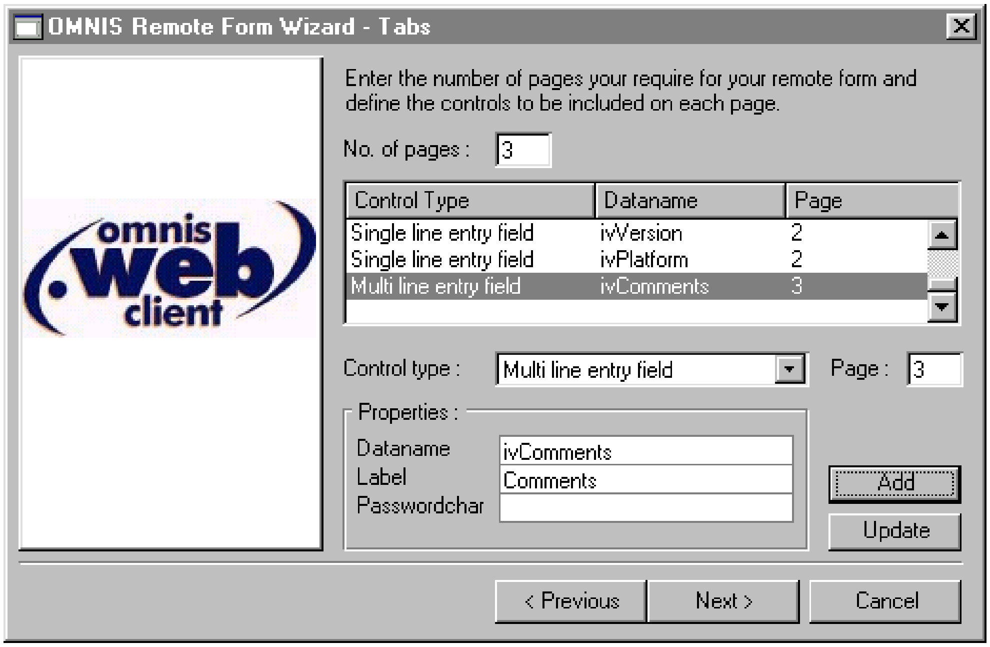
Properties :

Dataname	Var1
Label	Label
Passwordchar	

- Enter the number of pages for your form
- Select the control type for the first field on the first page
- Enter the dataname and text label for the first field and click on the Add button
- Select the control type, enter the dataname and text label for the second field on the first page, and click on the Add button
- Add the remaining fields for the first page, then select the second page and add fields as above

The list in the wizard window shows the fields you have added to each page in the form. At any stage you can click on a field in the list, edit its details, and click on the Update button.

The following screenshot shows a wizard that defines 3 pages, with a few fields on the first two pages, and a single multi-line field on the third page.



Enter the number of pages your require for your remote form and define the controls to be included on each page.

No. of pages : 3

Control Type	Dataname	Page
Single line entry field	ivVersion	2
Single line entry field	ivPlatform	2
Multi line entry field	ivComments	3

Control type : Multi line entry field   Page : 3

Properties :

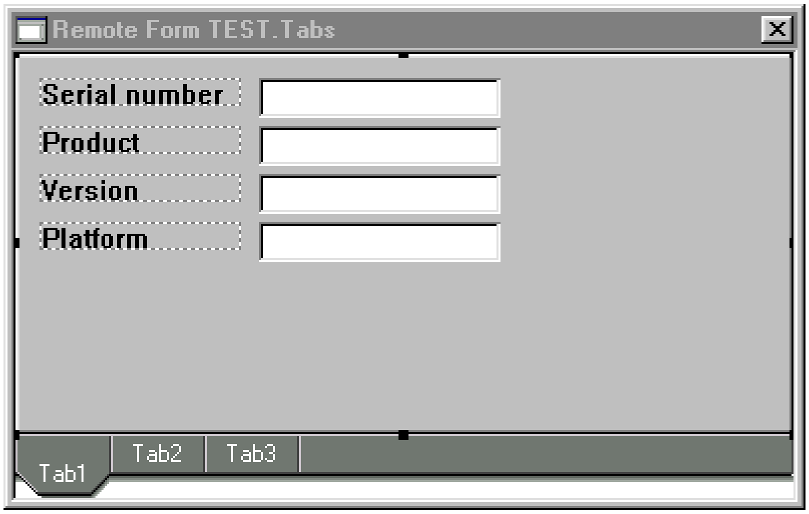
Dataname	ivComments
Label	Comments
Passwordchar	

- When the fields for each page in your tabbed form are complete, click on the **Next** button
- Select the type of web browser and click on the **Next** button

The wizard now creates the remote form using the details you have entered. Note that OMNIS creates a remote task as well.

- To modify the form, double-click on it in the Class Browser

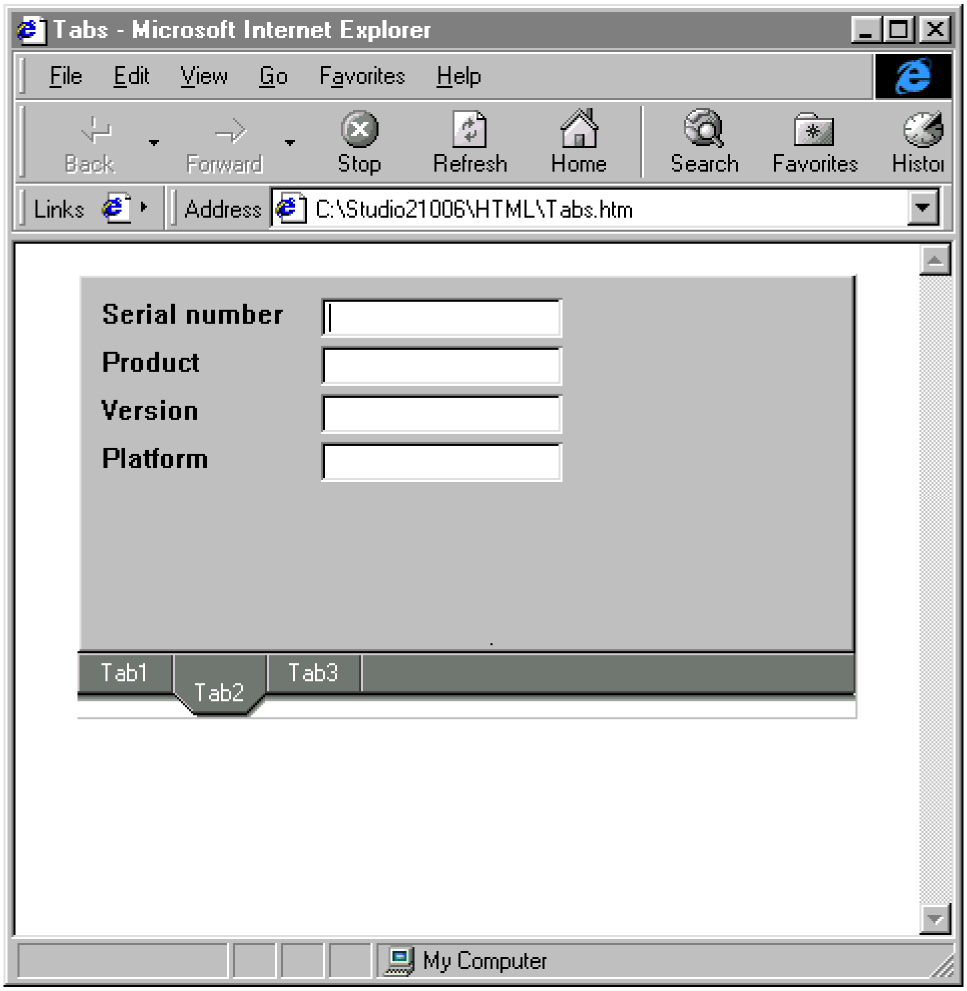
To examine the contents of each page in design mode, you need to click on the page pane field (click on the gray background, away from the fields and text labels), open the Property Manager or bring it to the top by pressing F6/Cmnd-6, and edit the **currentpage** property. For example, to look at the second page, click on the page pane field, enter 2 in the **currentpage** property in the Property Manager, and press Return (note the tab strip at the bottom of the window will not change in design mode). You should see something like the following screenshot.



To test the form

- Press Ctrl/Cmnd-T, or Right-click/Ctrl-click on the form and select the **Test Form** option from the context menu

When you press Ctrl/Cmnd-T, OMNIS opens your browser automatically and displays the form.


Tabs - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History
Links Address C:\Studio21006\HTML\Tabs.htm
Serial number
Product
Version
Platform
Tab1 Tab2 Tab3
My Computer

The form is opened in your web browser and you can click on the tabs to change the current page on the form. At this stage your form does not handle the data you type into the form, but this gives you an idea of how the form will appear to the end user.

Note that the browser uses a template html page created for you by the form wizard. The template page is placed in the **html** folder in the main OMNIS folder. This html page has the OMNIS Web Client embedded in it by default with all the correct parameters set up automatically.

If your web browser does not open you should check that the OMNIS Web Client is installed and registered properly.

- Close or minimize the browser when you want to return to OMNIS

Wizard Form Wizard

The Wizard form wizard creates a remote form containing a page pane control, and Next, Previous and Finish buttons. The resulting form is rather like a wizard in that it allows the user to enter data on a number of panes using the Next button, and gives you more control over the order in which the user is prompted for information. In the wizard you can add fields and controls to each pane in the form. When you have created the form you need to program the Finish button to process the contents of the form.

To create a Wizard Form

- Open the Component Store and click on the Remote Form Classes button
- Select the Wizard remote form wizard and drag its icon onto your open library in the OMNIS Class Browser
- Name the new remote form and press Return
- Select an existing remote task from the list of tasks provided, and click on the Next button

or

- Click on the Create New Task option, select a template from the list provided such as the Plain Remote Task, name the new remote task, and click on the Next button
- Enter the number of pages for your form
- Enter the details for each field on the first page, including control type, dataname and text label, clicking on the Add button for each one
- Add the remaining fields for the other pages in your form

The list in the wizard window shows the fields you have added to each page in the form. At any stage you can click on a field in the list, edit its details, and click on the Update button. You can delete a field in the list by Right/Ctrl-clicking on the field and selecting Delete Field from its context menu.

- When the fields for each page in your wizard-style form are complete, click on the **Next** button
- Select the type of web browser and click on the **Next** button

The wizard now creates the remote form using the details you have entered. Note that OMNIS creates a remote task as well.

- To modify the form, double-click on it in the Class Browser

To test the form

- Press Ctrl/Cmnd-T, or Right-click/Ctrl-click on the form and select the **Test Form** option from the context menu

Submit Form Wizard

The Submit form wizard creates a standard submit form containing your fields, Submit and Clear buttons. The wizard lets you add fields and controls to the form. In the resulting form, you need to program the Submit button to process the contents of the form.

To create a Submit Form

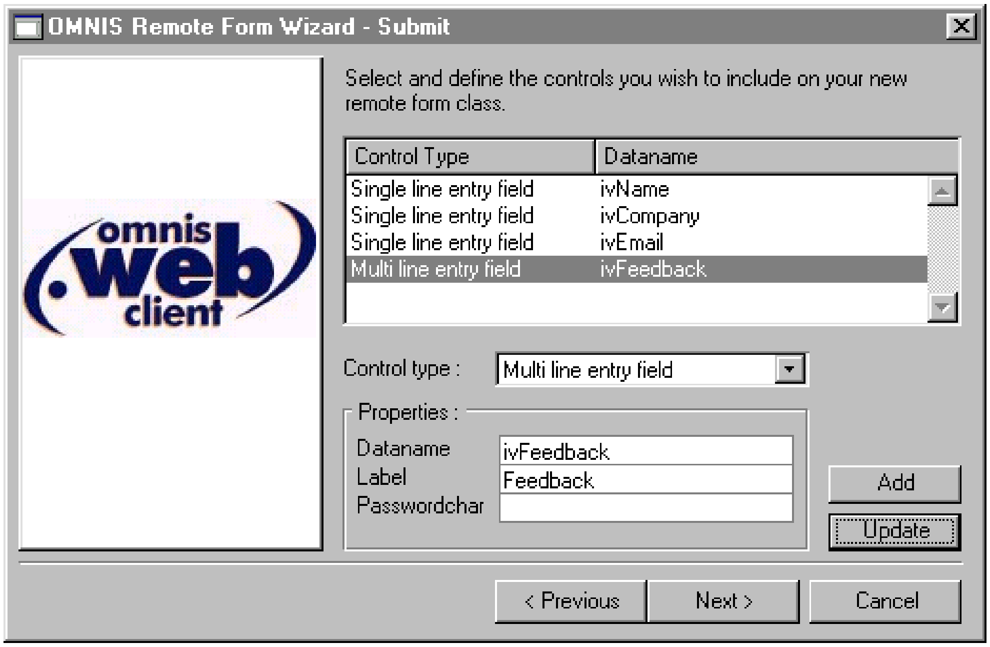
- Open the Component Store and click on the Remote Form Classes button
- Select the Submit remote form wizard and drag its icon onto your open library in the OMNIS Class Browser
- Name the new remote form and press Return
- Select an existing remote task from the list of tasks provided, and click on the Next button

or

- Click on the Create New Task option, select a template from the list provided such as the Plain Remote Task, name the new remote task, and click on the Next button
- Enter the details for each field on the form, including control type, dataname and text label
- Use the Add button to add each field in the form

The list in the wizard window shows the fields you have added to the form. At any stage you can click on a field in the list, edit its details, and click on the Update button.

The following screenshot shows a Submit wizard that defines a number of fields for a typical feedback-type form, including single-line entry fields for Name, Company name, E-mail address, and a multi-line field for the Feedback comments.



Select and define the controls you wish to include on your new remote form class.

Control Type	Dataname
Single line entry field	ivName
Single line entry field	ivCompany
Single line entry field	ivEmail
Multi line entry field	ivFeedback

Control type : Multi line entry field

Properties :

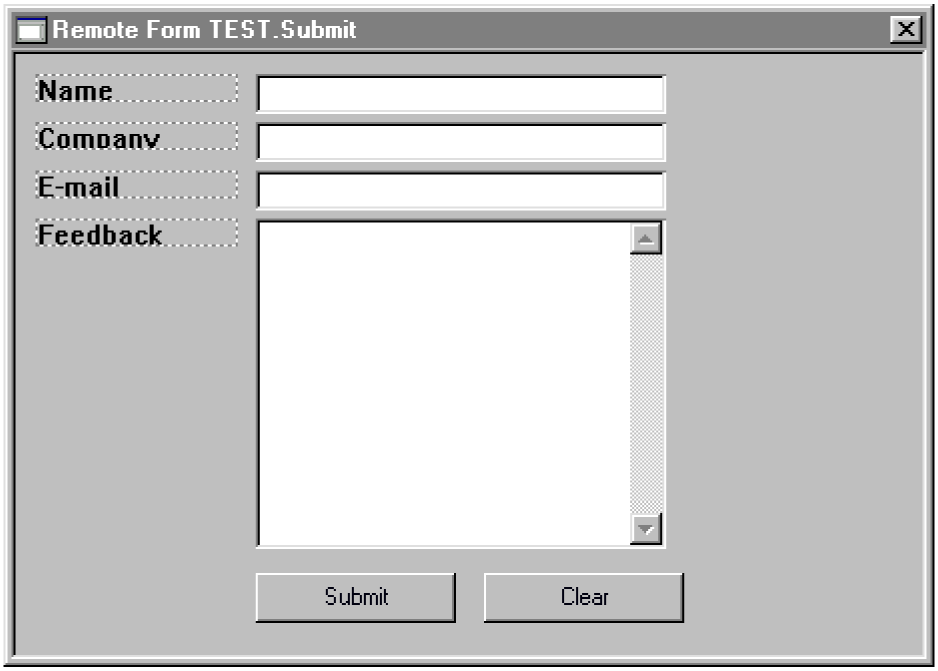
Dataname	ivFeedback
Label	Feedback
Passwordchar	

- When the details for your form are complete, click on the **Next** button

- Select the type of web browser and click on the **Next** button

The wizard now creates the remote form using the details you have entered. Note that OMNIS creates a remote task as well.

- To modify the form, double-click on it in the Class Browser



To test the form

- Press Ctrl/Cmnd-T, or Right-click/Ctrl-click on the form and select the **Test Form** option from the context menu

Password Form Wizard

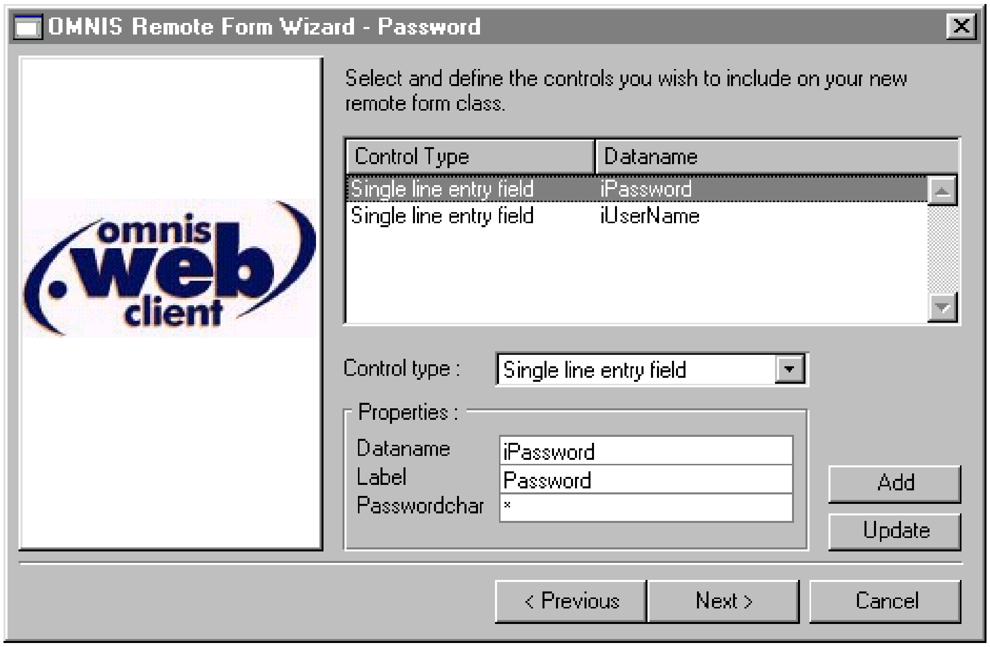
The Password form wizard creates a standard password form containing Password and Username fields. The wizard specifies these two fields by default, but you can change these or add your own fields. In the resulting form you need to program the Submit button to send the contents of the username and password fields to your server application.

To create a Password Form

- Open the Component Store and click on the Remote Form Classes button
- Select the Password remote form wizard and drag its icon onto your open library in the OMNIS Class Browser
- Name the new remote form and press Return
- Select an existing remote task from the list of tasks provided, and click on the Next button

or

- Click on the Create New Task option, select a template from the list provided such as the Plain Remote Task, name the new remote task, and click on the Next button



Select and define the controls you wish to include on your new remote form class.

Control Type	Dataname
Single line entry field	iPassword
Single line entry field	iUserName

Control type : Single line entry field

Dataname	iPassword
Label	Password
Passwordchar	*

- If you like, you can change the Password and Username field details using the Update button, or add some more fields as required

- When the details for your form are complete, click on the **Next** button

- Select the type of web browser and click on the **Next** button

The wizard now creates the remote form using the details you have entered. Note that OMNIS creates a remote task as well.

- To modify the form, double-click on it in the Class Browser

To test the form

- Press Ctrl/Cmnd-T, or Right-click/Ctrl-click on the form and select the **Test Form** option from the context menu

Sidebar Form Wizard

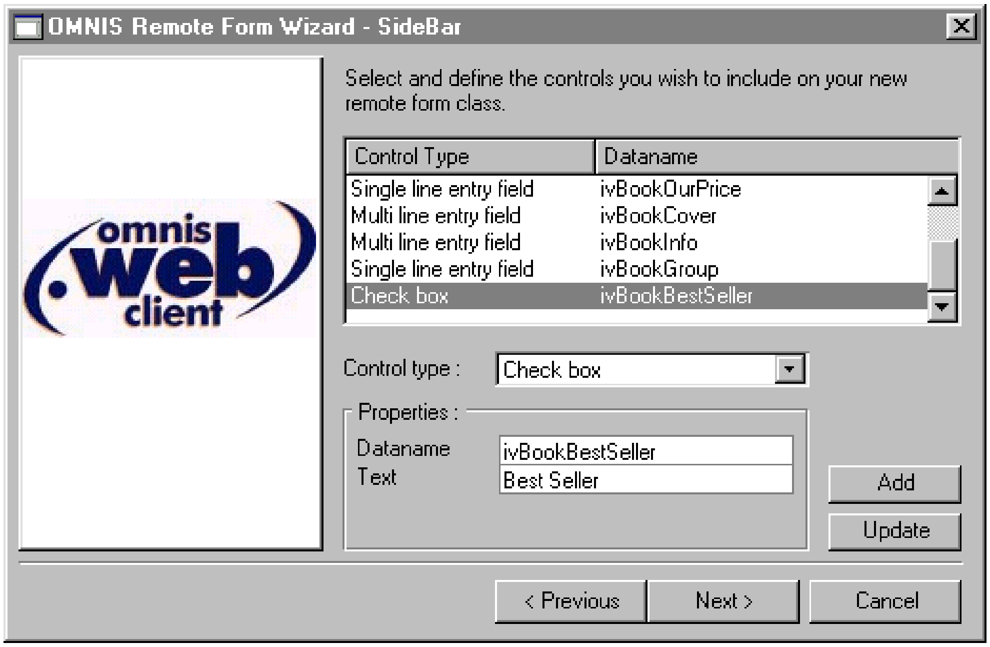
The Sidebar form wizard creates a form containing a number of fields and a sidebar control. In the wizard you can specify the groups and group items for the Sidebar control. The wizard creates a list containing the group items in your sidebar. When you have created your form you will have to add some programming behind the sidebar to allow it do respond to clicks or double-clicks.

To create a Sidebar Form

- Open the Component Store and click on the Remote Form Classes button
- Select the Sidebar remote form wizard and drag its icon onto your library in the OMNIS Class Browser
- Name the new remote form and press Return
- Select an existing remote task from the list of tasks provided, and click on the Next button

or

- Click on the Create New Task option, select a template from the list provided such as the Plain Remote Task, name the new remote task, and click on the Next button



Select and define the controls you wish to include on your new remote form class.

Control Type	Dataname
Single line entry field	ivBookOurPrice
Multi line entry field	ivBookCover
Multi line entry field	ivBookInfo
Single line entry field	ivBookGroup
Check box	ivBookBestSeller

Control type : Check box

Properties :

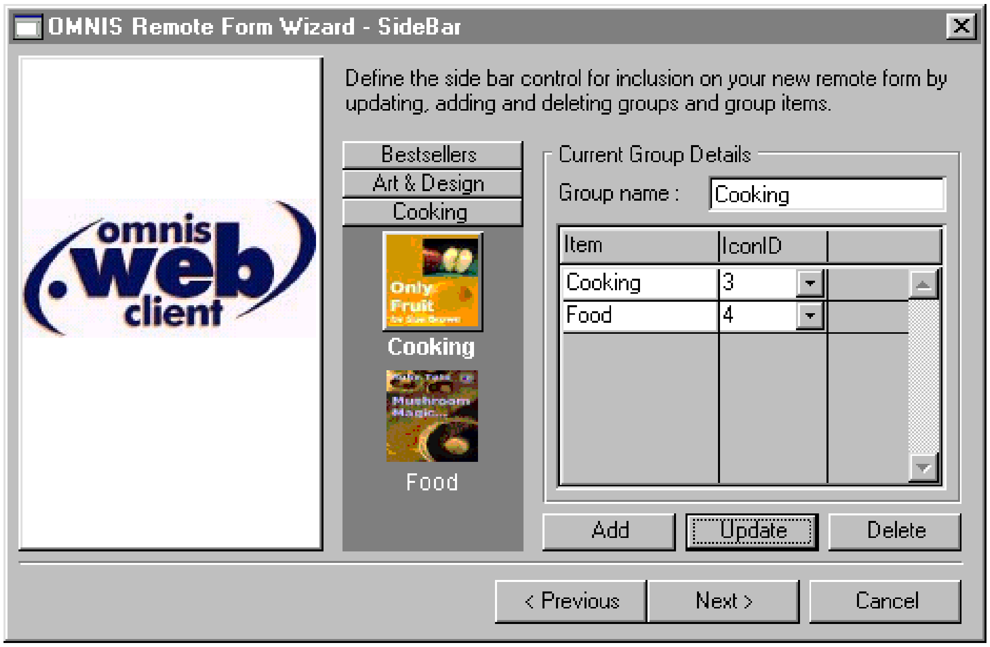
Dataname ivBookBestSeller

Text Best Seller

- Enter the details of the fields you require on the form, and click on the Next button

For the Sidebar form wizard you need to enter the different groups and separate group details. The wizard lets you enter the group name, and the name and icon id for each item in a group. When you click in the IconID field the Select Icon dialog opens which lets you select an icon from the #ICONS system table in the current library, or the OMNISPIC or USERPIC icon data file.

The following screenshot shows three groups and a couple of items in each group.



Define the side bar control for inclusion on your new remote form by updating, adding and deleting groups and group items.

Group name : Cooking

Item	IconID
Cooking	3
Food	4

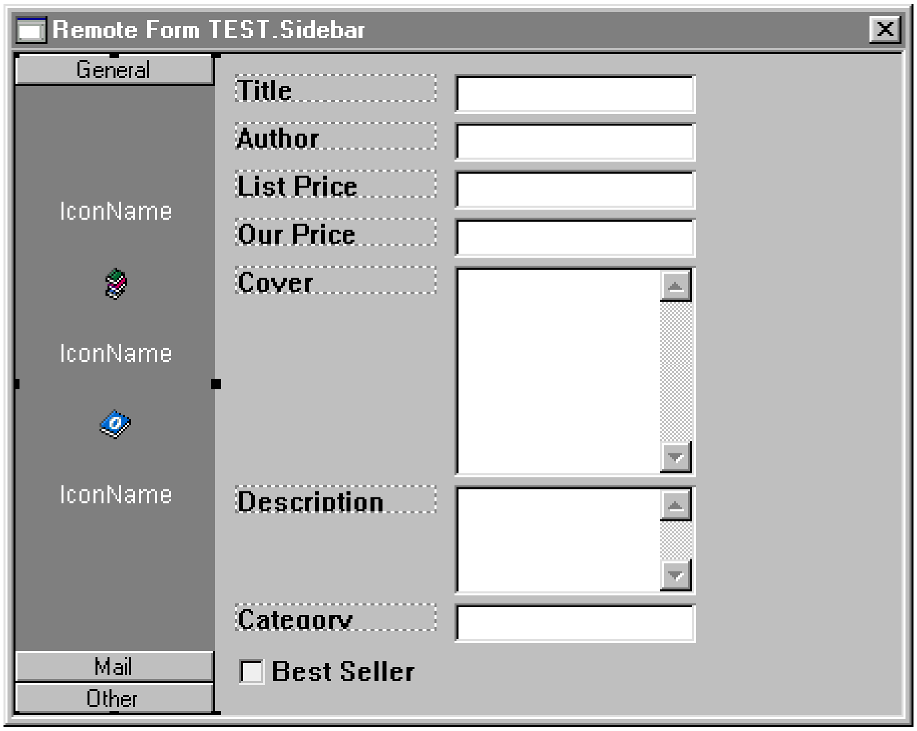
- When your Group details for the form are complete, click on the **Next** button

- Select the type of web browser and click on the **Next** button

The wizard now creates the remote form using the details you have entered. Note that OMNIS creates a remote task as well.

- To modify the form, double-click on it in the Class Browser

The following screenshot shows the form created using the details shown above. Note that in design mode your sidebar will not display the group or categories you have just entered in the wizard. The details for the sidebar are retrieved from a list contained in the form and only created at runtime.



To test the form

- Press Ctrl/Cmnd-T, or Right-click/Ctrl-click on the form and select the **Test Form** option from the context menu

Back in design mode you need to add a method behind the Sidebar control to respond to clicks. For example, you could add the following method to load the list line number of the item clicked and then execute a method using this information.

```
On evIconPicked
   Set current list iSideBarList  ;; the list behind the sidebar
   Load from list {pLinenum}  ;; loads linenumber of selected item
   Do ...something
```

Sidebar with pages Form Wizard

The Sidebar with pages form wizard creates a form containing a page pane and a sidebar control. In the wizard you can specify the groups and group items for the Sidebar control. The wizard creates a list containing the group items in your sidebar. The wizard lets you specify the fields for each pane and allows you to link each pane to a particular group in the sidebar control.

To create a Sidebar with pages Form

- Open the Component Store and click on the Remote Form Classes button
- Select the **Sidebar with pages** remote form wizard and drag its icon onto your library in the OMNIS Class Browser
- Name the new remote form and press Return
- Select an existing remote task from the list of tasks provided, and click on the Next button

or

- Click on the Create New Task option, select a template from the list provided such as the Plain Remote Task, name the new remote task, and click on the Next button
- Specify the number of pages you require on the form, and enter the details for the fields you want on each page, then click on the Next button
- Next you specify the group details for your Sidebar
- When the details for your form are complete, click on the **Next** button
- Select the type of web browser and click on the **Next** button

The wizard now creates the remote form using the details you have entered. Note that OMNIS creates a remote task as well.

- To modify the form, double-click on it in the Class Browser

To test the form

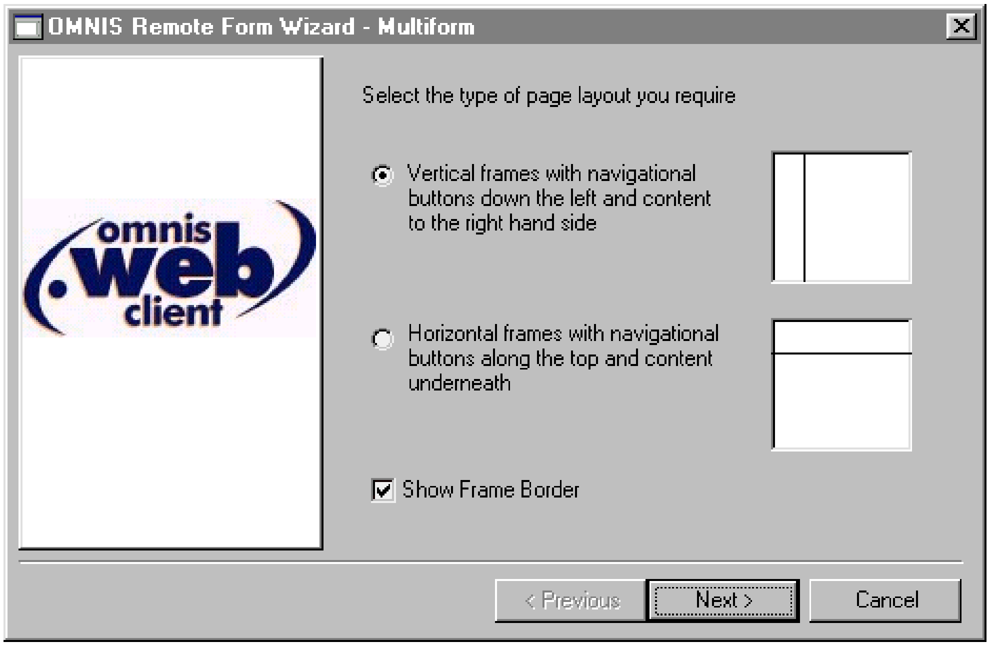
- Press Ctrl/Cmnd-T, or Right-click/Ctrl-click on the form and select the **Test Form** option from the context menu

Multiform Form Wizard

The Multiform wizard lets you create a complete website containing a home page, navigation bar and any number of OMNIS remote forms. The wizard creates an html page with a horizontal or vertical frame and navigation buttons linked to the different forms. In the wizard you can select existing remote forms to include in your website, or create new remote forms from the available wizards and templates.

To create a Multi Form website

- Open the Component Store and click on the Remote Form Classes button
- Select the Multiform remote form wizard and drag its icon onto your library in the OMNIS Class Browser
- Name the new remote form and press Return



OMNIS Remote Form Wizard - Multiform

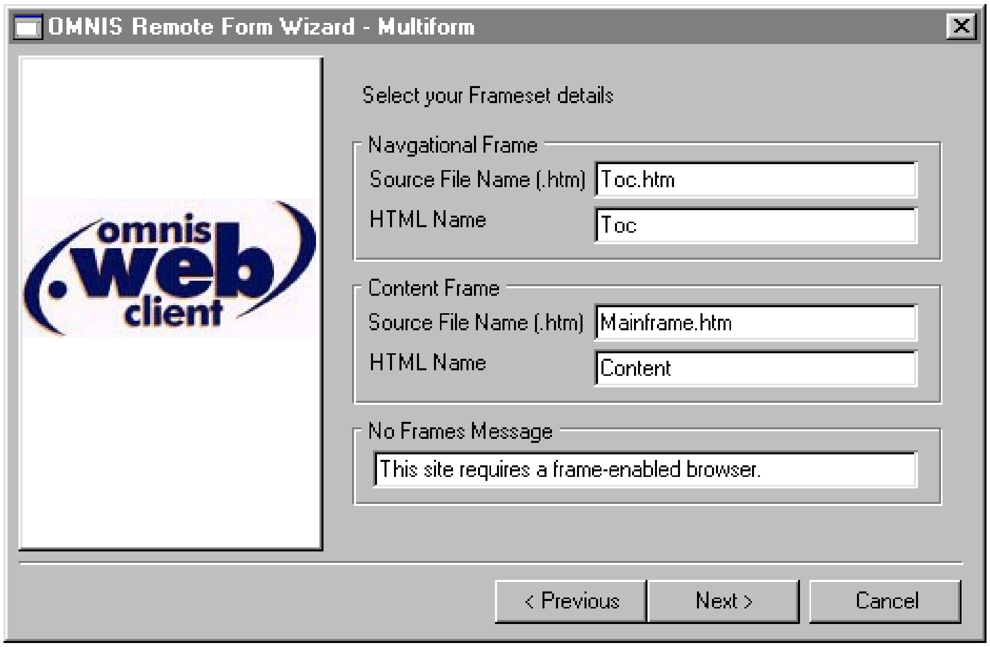
Select the type of page layout you require

[✓] Vertical frames with navigational buttons down the left and content to the right hand side

[ ] Horizontal frames with navigational buttons along the top and content underneath

[✓] Show Frame Border

- Select the layout for your html pages, either **Vertical** or **Horizontal** navigation bar, and uncheck the **Show frame border** option if you want to hide the frame border
- When you’ve selected the layout you require click on the **Next** button



Select your Frameset details

Navgational Frame

Source File Name (.htm)	Toc.htm
HTML Name	Toc

Content Frame

Source File Name (.htm)	Mainframe.htm
HTML Name	Content

No Frames Message

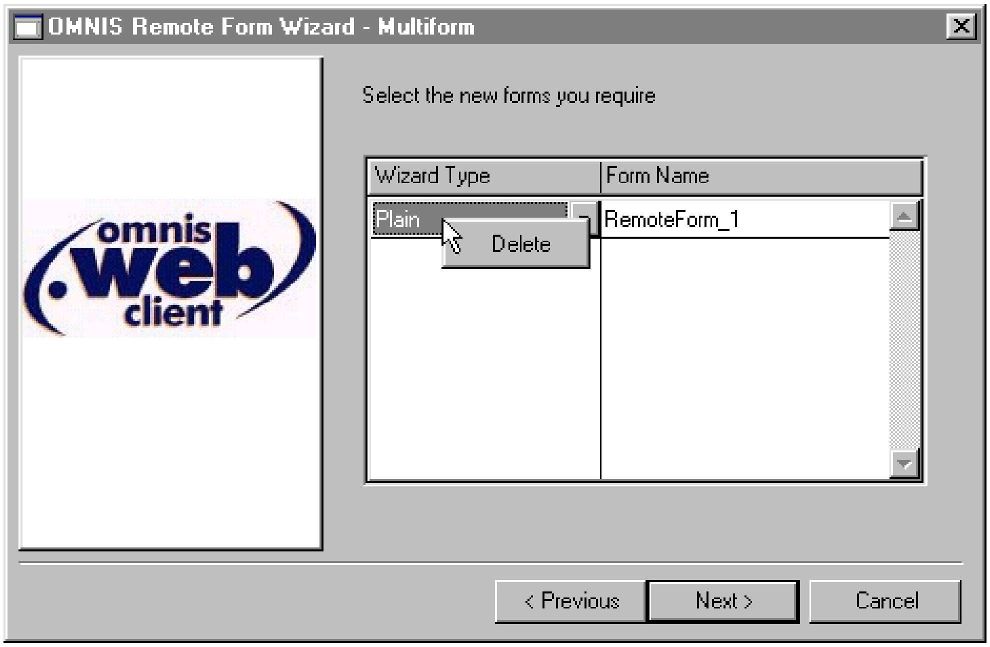
This site requires a frame-enabled browser.

Next the Multiform wizard prompts you to enter the Frameset details. You can enter the name of the html file for the navigation frame or table of contents, and the name of the content frame or Home page. These files are added to the HTML folder in the main OMNIS folder, and you can edit them at a later date.

- Enter the file names for the Navigational and Content frames or accept the default name and click on the **Next** button

Next the Multiform wizard lets you select any existing remote forms in your library that you want to include in your website. If your library does not contain any remote forms, click on **Next** to go the next stage straightaway. Otherwise

- Select the name of any existing remote forms in your library that you want to include in your website, and click on the **Next** button


OMNIS Remote Form Wizard - Multiform
Select the new forms you require
Wizard Type
Form Name
Plain
RemoteForm_1
Delete
< Previous
Next >
Cancel

The Multiform wizard now lets you enter a list of new remote forms you may want to add to your website. If your library does not contain any remote forms you will need to specify a list of new forms in the table, which lets you specify a Wizard type and Form name for each new remote form required.

If you don't want to add any new remote forms you can delete the contents of the list by Right/Ctrl-clicking on the list items and selecting the **Delete** option, as shown above.

- Enter a list of new remote forms you require or clear the list as above, then click on the **Next** button
- Select the type of web browser and click on the **Next** button

If you listed any new remote forms to be created, the appropriate remote form wizards are now launched. When you have completed the wizards for any new remote forms you are returned to the Class Browser. The Multiform wizard creates the specified html pages and adds any new remote forms to your library. To test your multi-form website you need to open the appropriate html file. To do this:

- Open the HTML folder in your main OMNIS folder
- Double-click on the **Frameset.htm** file to open it in your web browser

Debugging and Testing Remote Forms

In order to run and debug a remote form, the development version of OMNIS lets you open a remote form in your local web browser. You can press Ctrl/Cmnd-T while designing your remote form, or pick **Test Form** from the form's context menu, to display your form in your browser. The OMNIS Web Client establishes a remote connection to the debugging OMNIS. You can set breakpoints in the form code and step through the code as usual.

Creating a New Remote Form

You can create a remote form from scratch using the New Remote Form template available in the Component Store, or from the **Class>>New** option in the Class Browser. However this type of form does not contain any fields or instance variables, and you have to link it to a remote task manually by specifying its $designtaskname property.

To create a new remote form

- Open the Component Store and drag the New Remote Form template onto your library in the OMNIS Class Browser
- Name the new remote form and press Return
- Double-click on the remote form in the Class Browser and add all the instance variables, fields, and controls as required

Remote Form Properties

Remote form classes have all the standard properties of an OMNIS class together with the **iconpages** property, shown under the General tab in the Property Manager.

Icon Pages

Some remote form components, such as buttons, can have icons. The icons for all the components on a remote form are sent to the client, together with the class data, when the form is instantiated. However, OMNIS does not send individual icons to the client, but complete pages, so you should use icons from as few icon pages as possible. The **iconpages** property for a form specifies a list of icon pages which are to be sent to the client. The icons for the objects in a remote form can be stored in the #ICONS system table in the current library, or the OMNISPIC or USERPIC icon datafile.

In the Property Manager you can click on the droplist for the **iconpages** property and select a page or a number of icon pages. The names of icon pages are stored in the **iconpages** property separated by commas, and different pages from the #ICONS system table, the

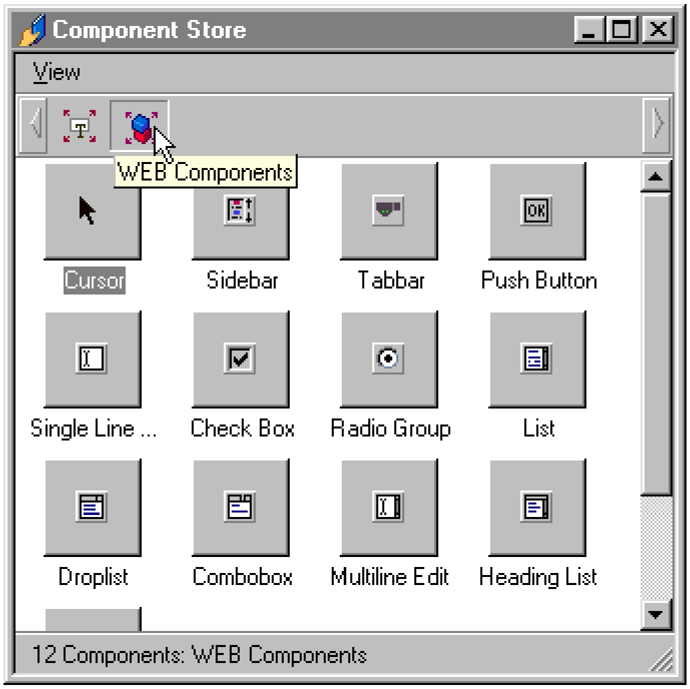
OMNISPIC, and USERPIC icon data files (in that order) are further separated by colons. For example, two pages called Books and General from the #ICONS table in the current library and one page called Embedded Colors from USERPIC datafile are stored as: Books,General;;Embedded Colors.

Remote Form Controls

The remote form wizards add controls to your forms automatically, but if you are creating your own forms or adding fields to an existing form, the OMNIS Component Store lets you create many types of control. When you have created a control in your remote form you can modify it by changing its properties in the Property Manager.

To create a remote form control from the Component Store

- Open your remote form in design mode
- Press F3/Cmnd-3 to open the Component Store or to bring it to the top
- Click on the WEB Components button in the Component Store; note Label objects, Border objects, and Page panes are in the WEB Background Objects group in the Component Store, selected by default



- Drag the required control type from the Component Store and drop it onto your remote form

or, to draw a form control of a particular size

- Select the required control type in the Component Store by clicking on its icon
- Click and drag on your remote form to define the size of control you want

or, to place a control in your form automatically

- Double-click on the appropriate icon in the Component Store

When you double-click on an icon in the Component Store a control of that type will appear in the center of your remote form. You can repeat this as many times as you want to place multiple copies of the same type of form control.

For further details about creating fields and controls using the Component Store and modifying their properties via the Property Manager, see the *Using OMNIS Studio* manual.

The following web components are available in the Component Store.

Labels

Standard text object for labeling other form objects. You can enter or edit the text for the object in its **text** property.

Border Objects

Rectangle with various border styles. Under the Custom tab in the Property Manager, you can set the border style for this object by specifying the **outer** and **inner** properties, and the **bordergap**.

Pushbuttons

Button control for executing a method when the user clicks on the object. You can enter or edit the text for the button in its **text** property. Note a button does not have a **dataname**. You can enable specific events for the button by selecting the appropriate events in the **events** property for the object. For example, you can specify **evClick** so the button responds to single clicks. When the event is triggered by the click the method behind the object is executed. See the *Events* section later in this chapter.

Single Line Entry Fields

Single-line control for entering or displaying character data. The single-most important property of an entry field is its **dataname**, that is, the name of the instance variable the field uses to display or insert its data into. If you created your remote form using one of the form wizards, the **dataname** property of each field is set up for you automatically, directly linking each form field to a specific column or data field in your database. Otherwise, the instance variable for an entry field should be of type **Character** or **National** and its name should be entered in the **dataname** property. You can enable an **evBefore** or **evAfter** event for an entry field in its **events** property; these events detect when the user enters or leaves the field and lets you write a method to intercept these events. See the *Events* section later in this chapter.

Date and Time Display Format

Single- and multi-line edit fields have the **displayformat** property that controls how date & time variables are displayed on the client machine. It can be one of a number of different formats: None, Time, ShortDate, ShortDateTime, LongDate, and LongDateTime. The way these are displayed on the client will entirely depend on the users configuration set in the Control Panel>>Regional Settings (under Windows).

Multi Line Entry Fields

Multi-line field control with vertical scrollbar for entering or displaying large character data. The properties and events for a multi-line field are the same as a single-line field; see above.

Check Boxes

Control that the user can check or uncheck. If the user checks the box the variable behind the field has a value of 1, otherwise it has a value of zero. The instance variable for a check box should be of type **Boolean** and its name should be entered in the **dataname** property. You can enter or edit the text for the check box in its **text** property.

Radio Group Objects

Toggle field that allows the user to select one of a number of mutually exclusive choices by clicking on a radio button. You can enter or edit the text for each radio button in the group as a comma-separated list in the object’s **text** property. The instance variable for a Radio group should be of type **Number** to accept the number value of the selected button; the variable name should be entered in the **dataname** property. Under the Custom tab in the Property Manager, you can set the **columncount**, the **minvalue** and **maxvalue** for the group range, and you can specify whether or not the group has **horizontal** alignment.

List Boxes

Multi-line list control that displays the data in an OMNIS list variable and allows the user to click or double-click a line to make a selection. The instance variable for a List control should be of type **List** and its name should be entered in the **dataname** property. You can enable specific events for the list control by selecting the appropriate events in the **events** property for the object. For example, you can specify **evDoubleClick** so the list responds to double-clicks. When the event is triggered by the double-click the method behind the list is executed. See the *Events* section later in this chapter.

Drop Lists

List control that drops down when clicked to display the data in an OMNIS list variable. The instance variable for a Drop list control should be of type **List** and its name should be entered in the **dataname** property.

Combo Boxes

Control that combines a dropdown list and entry field that allows the user to select from the list or type in a value; the control requires two variables, a character variable and an OMNIS list variable. The **dataname** property for a Combo box specifies the variable of type Character. Under the Custom tab in the Property Manager, the ::listname property specifies the variable of type List for the list part of the control. You can enable specific events for a Combo box control by selecting the appropriate events in the **events** property for the object. See the *Events* section later in this chapter.

Headed Lists

Multi-line list control that has button-style headers and adjustable column widths. The instance variable for a Headed list control should be of type **List** and its name should be entered in the **dataname** property. Under the Custom tab in the Property Manager, you can specify the number of columns in **colcount**, the **columnnames** for list headings, and **columnwidths** for the headed list. Like the simple list box, you can enable specific events for the headed list by selecting the appropriate events in the **events** property for the object.

Picture Objects

Graphical control for displaying picture data. The instance variable for a Picture field should be of type **Picture** and its name should be entered in the **dataname** property.

Sidebars

Selection bar that lets the user select an icon from a number of groups. The instance variable for a Sidebar control should be of type **List** and its name should be entered in the **dataname** property. The list variable for a sidebar stores static data representing the names and icon ids for the groups and group items in the sidebar control. You can build the list for a sidebar in the $construct() method of the form, using the $add() method and the following parameters:

```
Do ListName.$add('GroupName',IconID,'GroupItemName')
```

For example, the following method builds a list for a sidebar control that contains three groups (Cooking, Science, and Fiction) with two items in each group.

```
; declare iSideBarList of type List
Do iSideBarList.$define(iGroup,iIconId,iItem)
Do iSideBarList.$add('Cooking',0,'0')   ;; defines a group
Do iSideBarList.$add('Cooking',k48x48+3,'Cooking')  ;; group item
Do iSideBarList.$add('Cooking',k48x48+4,'Food') ;; group item
Do iSideBarList.$add('Science',0,'0')
Do iSideBarList.$add('Science',k48x48+5,'Science')
Do iSideBarList.$add('Science',k48x48+6,'Technology')
Do iSideBarList.$add('Fiction',0,'0')
Do iSideBarList.$add('Fiction',k48x48+7,'Fiction')
Do iSideBarList.$add('Fiction',k48x48+7,'Science Fiction')
```

To position a Sidebar control in the remote form you can set its **edgefloat** property; for example, you can set it to **kEFposnLeftToolBar** to “glue” the sidebar to the left side of the form. You can set further properties for the sidebar control, under the Sidebar and Text tabs in the Property Manager; to specify such things as a wash or tiled background for the sidebar.

Tab Bars

Selection strip that lets the user click on a tab, to change a page pane for example. Note a tab bar control does not have a **dataname**. You can set the properties for the tab bar under the TabBar tab in the Property Manager; for example, you can specify the **position** of the tab bar. You can enable specific events for a tab bar by selecting the appropriate events in the **events** property for the object. Specifically, **evClick** lets you detect which tab is selected, returning the tab number in the pLineNumber event parameter. For example, the following event handling method detects which tab is clicked on and sets the current page of a page pane (called Mainpage) in the form:

```
On evClick
  Do $cinst.$objs.Mainpage.$currentpage.$assign(pLineNumber)
```

Page Panes

Multi-page control that lets you add fields and text to each pane; this control should be used with the tab bar control to change the current pane. Under the General tab in the Property Manager, you can specify the **pagecount** and **currentpage** for the control. Note a page pane control does not have a **dataname** and reports no events. The following example shows the event handling method for a tab bar and shows how you can change the current pane of a page pane control. Note the remote form contains two page panes.

```
On evClick
  Do $cinst.$objs.Mainpage.$currentpage.$assign(pLineNumber)
  Switch pLineNumber    ;; the tab number clicked on
    Case 1
      Calculate $cinst.$objs.Subpage.$currentpage as 4
    Case 3
      Calculate $cinst.$objs.Subpage.$currentpage as 5
    Case 4
      Calculate $cinst.$objs.Subpage.$currentpage as 6
    Default
      Calculate $cinst.$objs.Subpage.$currentpage as 1
  End Switch
  Do $cinst.$senddata(#NULL)
```

Programming Remote Forms

When writing methods for a remote form class or its controls, you must consider that the class may be instantiated many times, depending of how many clients are connected simultaneously. Although you can use any OMNIS command or notation and create methods of any length, you must choose carefully what commands you use and how you program certain operations.

You cannot modify a remote form instance at runtime, that is, you cannot add objects to the form instance, or remove them. The same applies to instance variables. These type of changes must be made to the remote form class prior to opening an instance, if they are required. In addition, invalid notation will generate an error when executed.

Window related commands such as ‘Redraw’ will not work and should not be used. In this case use notation the $cobj.$redraw(), or ObjectName.$redraw().

No calls are made to the $control() method of a remote form class’ task. Events are sent only to the objects $event() method. There are no events generated for the remote form itself.

Optimizing Data Handling

When executing events on the server, by default OMNIS returns all instance variables to the client. To optimize what variable data is returned you can specifically state which data you wish to be returned. You can do this using the $cinst.$senddata(ivVar1,…ivVarN) method. Once $senddata() has been executed, OMNIS will only return the data of the specified variables. So, when you change the data for a control, normally it is enough to issue a $redraw for the control using $cobj.$redraw(). In remote forms you will also need to issue a $senddata(), specifying the dataname of the control. You may execute a $senddata() method more than once, and it’s ok to specify the same instance variable more than once.

If you issue a $cinst.$senddata() without specifying any instance variables, it will clear the send state of previously specified variables, and all variables will be sent, unless you issue another $senddata(). If you do issue another $senddata(), only the newly specified variables will be sent. The notation $cinst.$senddata(#NULL) stops OMNIS from sending any data. You can also use $cinst.$senddata(#NULL,ivVar1) to ensure that only ivVar1 is sent.

Icon pages

You can add icon pages using the notation RemoteFormRef.$iconpages.$add(iconID). OMNIS adds the name of the page containing the specified icon to the $iconpages property of the remote form.

Open windows and User Prompts

You should not use any commands or programming in your library that opens a message box or prompts the user in any way. For example, if you use the standard *OK message* command and it is executed, the message appears on the OMNIS server, not in the client's browser, and the application is effectively halted since the message is not cleared until you click OK; see below for method to show a message on the client. If a message or prompt remains open in the OMNIS server application all client to server interaction is suspended and the client's browser is locked up. This situation is not serious during development on your local machine, since you can switch to OMNIS and clear the message box or prompt, but for an application deployed on the web this would be disastrous and would severely inhibit useability!

There are many commands that you cannot use for the above reason, they include: *OK message, Yes/No message, Working message, Prompt for input, Prompt for data file,* and so on.

You also need to consider the result of using any command or notation that opens a window, installs a menu or toolbar, prints a report, and so on. These commands all do something in the server application, and the result of any such command is not visible in the client's browser. For example, if you run the *Open window* command, an OMNIS window class is opened in the server application and not in the client browser; this is Ok for a server admin window, for example, but is of no use for the client. Therefore, bear in mind that all user interaction over the web must be enabled using remote forms opened in the client.

OK Messages

You can display a message on the client using the $showmessage() method. For example, the following code for a pushbutton opens a message dialog.

```
On evClick   ;; button must have evClick enabled
  Do $cinst.$showmessage('Message','Title')
```

Smart lists

The OMNIS Web Client and Remote forms do not support OMNIS smart lists.

Remote Form Instances

Once the OMNIS Web Client is connected to OMNIS on the server, the user can tab through the fields of the form and switch between its pages, entering data, clicking on buttons, selecting options, and so on. When the user causes an event, which has been enabled, the event is executed on the server. Only enabled events cause communications between the client and the server, all other events are ignored.

There is a group called $iremoteforms in $root. This group contains the chain of Remote Form Instances. On the server, only the non-visual part (mainly its methods) of the form is instantiated. The visual part of a form instance is instantiated on the client.

The $construct method of a remote form class can specify a parameter variable of type Row as its first parameter so it can access the parameters from the ActiveX. When the $construct of the form terminates, the forms' data, the instance variables, and all property changes made to the form and its components during construct, are returned to the client.

Form Caching

Server Form Caching

In order to improve performance at connection time, the OMNIS Runtime Server caches various items when the first client of a remote form connects. Subsequent connections will return this cached data. The following items are cached: the class data, the instance variables definition, the icon pages, and the font tables.

You can implement some sort of update process while the system is live, for example, the remote form has been changed, but the cache needs to be cleared. If the cache isn't cleared new connections will continue to return the old class data.

You can use the $root method $clearcachedforms() in your application to clear all the cached items on the OMNIS server.

Client Form Caching

In addition to the server form caching, the OMNIS Web Client caches any remote forms it has downloaded from the server on the clients hard disk. The client caches the remote form class data, the instance variable definitions, icon pages, and font tables. The client cache stores the modification dates of each item. When the client connects to the form at a later date, it passes these modification dates to the server, and the server returns only those items that have changed.

IMPORTANT NOTE: The icon pages receive their modified data from the #ICONS system table in the current OMNIS library. If you include icons from the OMNISPIC or USERPIC icon datafile and you update them, you must modify the #ICONS class to force all icons on the client to be updated; there is no other way for the OMNIS server to tell if icons in OMNISPIC.DF1 or USERPIC.DF1 have changed, other than checking the modification date of #ICONS.

Events

Remote forms do not generate any events themselves, only the $construct method of the form is called when a form is instantiated. Most of the available remote form objects do, however, generate events, although all events are disabled by default. You can enable specific events for an object by setting its **events** property in the Property Manager. When enabled, events are reported to the $event() method for the object. You can add some programming to the $event() method for an object, which is called when the event is triggered. For general information about handling events, see the *OMNIS Programming* manual.

Most remote form objects report the **evBefore** and **evAfter** events, which are triggered when the user is about to enter the field or leave the field, respectively. You can use the *On* command to detect events in your event handling methods. For example, in the $event() method for a single-line entry field you could use the following commands to detect the evBefore or evAfter event.

```
On evBefore
   ; do something..
On evAfter
   ; do something else..
```

Pushbuttons and all the list type controls report the **evClick** and **evDoubleClick** events, as well as evBefore and evAfter. The Booklist (heading list type) in the tutorial library has the following $event() method, which is executed when the user double-clicks on a line in the list:

```
On evDoubleClick
   Do method $findnow
      ; executes the $findnow method contained in the form.
      ; the method loads the details for the selected book and
      ; redraws the fields on the form
```

Other form controls have their own particular events, for example, the Sidebar control reports the **evIconPicked, evSetPicked,** and **evSetBeingPicked**; the first one is reported when an icon is selected, the other two are reported when the user selects an icon strip or group. The Sidebar control used in the tutorial library has the following $event() method, which is executed when the user selects an icon:

```
On evIconPicked
   Set current list iSideBarList
      ; makes the list var behind the sidebar the current list
   Load from list {pLinenum}
      ; loads the line from the list using the line number
      ; pLinenum param is sent to the method
   Do method $ctask.$buildbooks (iBookGroup)
      ; executes the $buildbooks method in the current task using
      ; the current value of iBookGroup from the selected list line
   Calculate iBookList as tBookList
      ; transfers the data from tBookList to iBookList instance var
      ; contained in the remote form
   Do $cinst.$objs.BookList.$redraw()
      ; redraws the Booklist field on the form
   Do $cinst.$senddata(iBookList)
      ; sends the iBookList data to the client
```

Chapter 4–Remote Task Classes

A *remote task* is a type of OMNIS class that handles the connection between a remote form and the OMNIS server application, and in some cases performs some server-side event handling and processing. Therefore, all remote forms require a remote task. When you create a remote form using one of the form wizards, OMNIS creates a remote task class for you automatically, so in most cases you can use this one. If you want to create your own remote task classes the Component Store contains a number of templates and wizards.

As an alternative to using the OMNIS Web Client and remote forms, OMNIS Studio lets you interact with your OMNIS application and database over the Internet using standard html forms and remote tasks. In this case, your html forms connect to an OMNIS remote task class direct, and no remote forms are required for this type of interaction. See the later in this chapter for more details about using standard html forms for direct access.

Creating Remote Task Classes using Wizards

You can create an OMNIS remote task from scratch or using one of the templates or wizards provided in the Component Store. The following templates are available:

– **Plain Remote Task**
  creates an empty remote task containing an $event method with code for evBusy and evIdle events

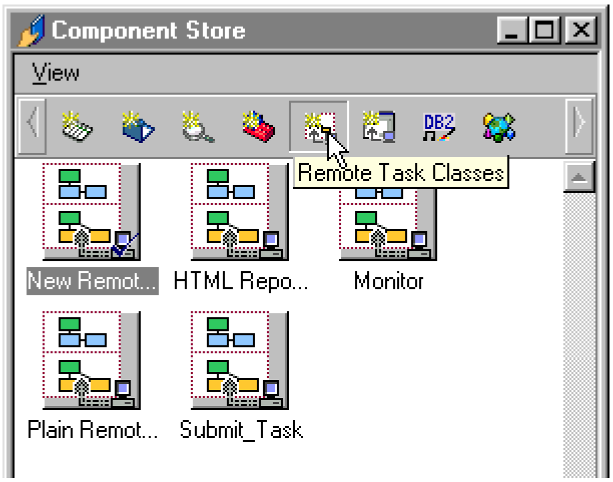
– **Monitor**
  creates a task and window to monitor remote connections.

– **HTML Report**
  creates a task to generate html reports on the fly

– **Submit**
  creates a task and html file containing a submit form which interacts with OMNIS direct.

To create a remote task class using a wizard

- Open your library in the OMNIS Class Browser
- Display the classes in your library using the View>>Down One Level menu option on the Browser menu bar
- Press F3/Cmnd-3 to open the Component Store or bring it to the top
- Scroll the top toolbar in the Component Store and click on the **Remote Task Classes** button to show the remote task wizards and templates



- Drag the wizard onto your library in the Class Browser
- Name the new class and press Return
- Follow the instructions in the wizard

Plain Remote Task Wizard

The Plain Task wizard creates a basic template remote task that is suitable for linking to most simple remote forms. The Plain remote task also has an $event() method containing a template event handling method that detects evBusy and evIdle events in the task. You can add your own code to handle these events.

The Plain remote task has a $construct() method containing a parameter variable called *pParams* of type *Row Variable*. This row variable can receive all the parameters of the ActiveX/plug-in, including the remote form name, task name, etc, and up to nine additional parameters which are embedded in the html page containing the form.

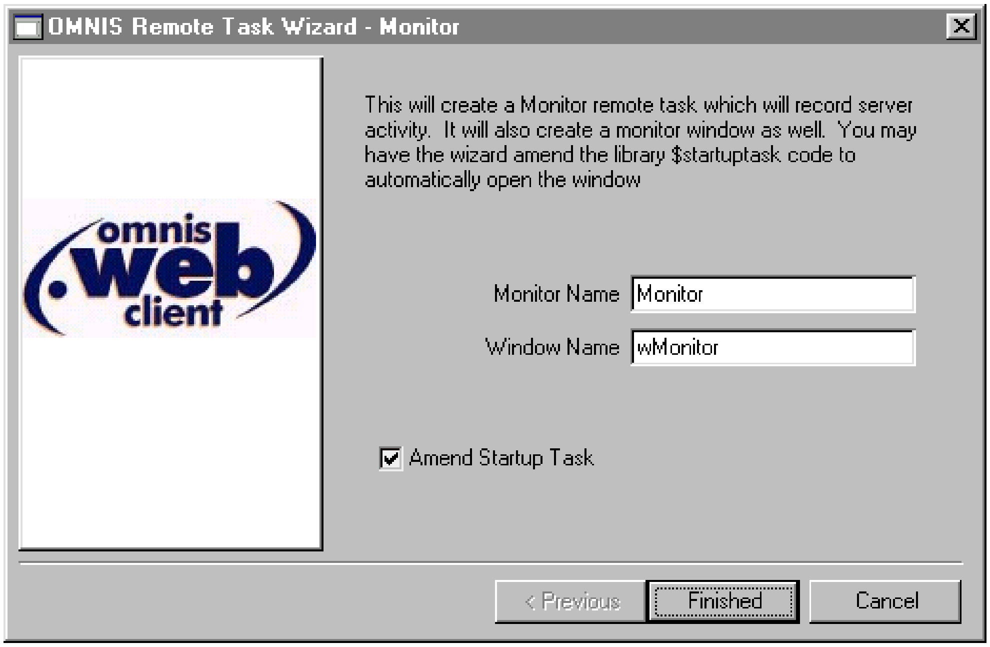
When you create a task using the Plain Task wizard you can specify the **Inherit from Monitor task** option. This option adds a set of “monitor” classes to your library which allows you to record client connections associated with the new plain task you are adding to your library. If you check the Monitor option, the wizard prompts you for details about the new monitor task. If your library does not contain a monitor task you need to specify the **Create New Monitor Task** option. If, however, your library contains a monitor task you can specify the **Use Existing Monitor Task** option to add the new plain task you are currently adding to your library to the existing monitor. See the *Monitor Wizard* section for further details.

Monitor Remote Task Wizard

The Monitor wizard creates a number of “monitor” classes, including a new task and monitor window, that allow you to record remote connections between clients using the OMNIS Web Client and the OMNIS server.

To create a Monitor window and associated classes

- Display the classes in your library using the View>>Down One Level menu option on the Class Browser menu bar
- Press F3/Cmnd-3 to open the Component Store or bring it to the top
- Scroll the top toolbar in the Component Store and click on the **Remote Task Classes** button to show the remote task wizards and templates
- Drag the **Monitor** wizard onto your library in the Class Browser
- Name the new class, or keep the default name, and press Return
- Follow the instructions in the wizard


OMNIS Remote Task Wizard - Monitor

This will create a Monitor remote task which will record server activity. It will also create a monitor window as well. You may have the wizard amend the library $startuptask code to automatically open the window

Monitor Name: Monitor

Window Name: wMonitor

[✓] Amend Startup Task

The wizard prompts you to enter the name of the new Monitor remote task and window, in most cases however, you can accept the default names. The **Amend Startup Task** option lets you add code to the Startup_Task in the current library to open the Monitor window at startup; this is checked by default, and in most cases you should let OMNIS change your startup code.

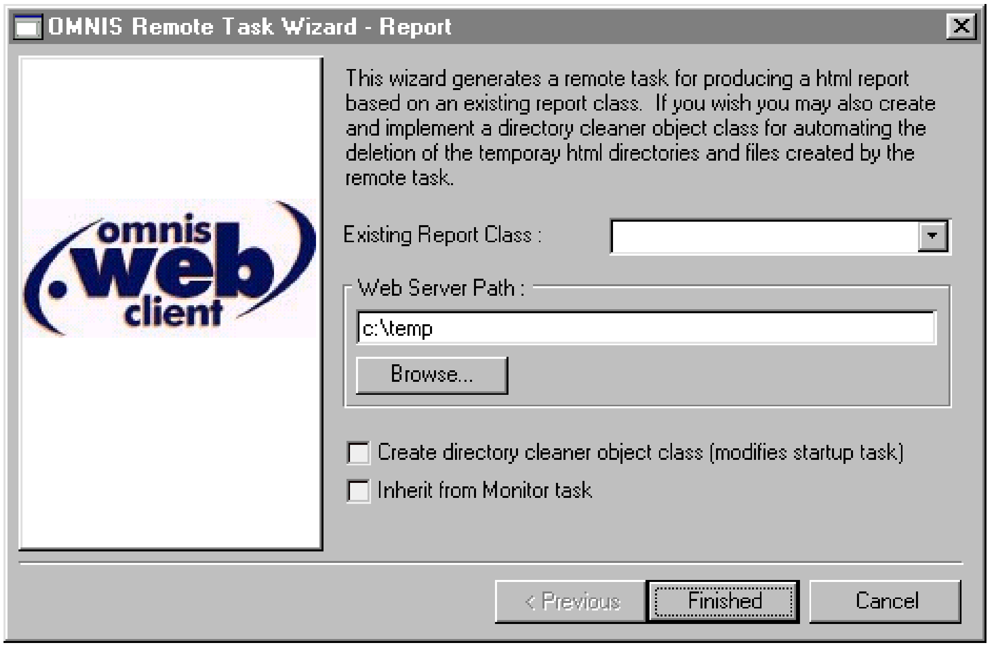
Monitor Window

The Monitor window, called wMonitor by default, has three panes. The **Connections** pane shows the connections grouped by remote form name. The **History** and **Server Usage** panes let you monitor the traffic flow on your OMNIS server and provide some general information about server usage. You can print the server usage using the **Print Report** button.

HTML Report Task Wizard

The HTML report task wizard creates a remote task that generates an Html report on-the-fly and returns it to the client. The $construct() method contains code to print a report to html using the temp directory object, and return an URL or error message to the client.

The wizard lets you base your Html report on an existing OMNIS report class. You can also specify the folder where the temporary report pages are placed, and you can add an object class to your library to clean out the contents of the temporary folder. In addition, you can add the new HTML report task to the Monitor task.


OMNIS Remote Task Wizard - Report

This wizard generates a remote task for producing a html report based on an existing report class. If you wish you may also create and implement a directory cleaner object class for automating the deletion of the temporay html directories and files created by the remote task.

Existing Report Class :

Web Server Path : c:\temp

Browse...
☐ Create directory cleaner object class (modifies startup task)
☐ Inherit from Monitor task
< Previous
Finished
Cancel

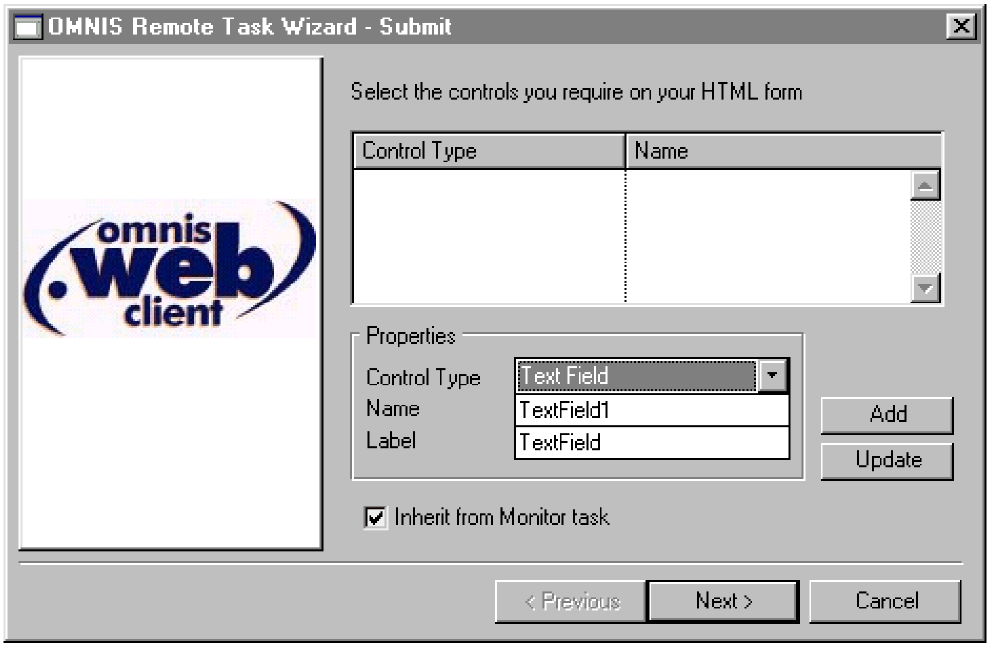
The HTML Report wizard creates an html file containing a Print Report button, using the name TaskName.htm, and places it in the OMNIS/HTML folder. When the button on the html form is pressed, the $construct() of the task creates a new html page in the webserver\omnishtml\[reportname] folder which contains the report. This page is then returned to the user. The Cleaner object sweeps the omnishtml folder in your webserver and deletes them every 1 minute.

Submit Remote Task Wizard

The Submit wizard creates a remote task and html file containing a submit form. The html form created by this wizard allows direct access to your database, and does not use a remote form or the OMNIS Web Client. See the later in this chapter for more details about using standard html forms for direct access.

The $construct() method in the Submit task contains variables to handle the data from the form in the associated html file. The wizard places the Submit html file in your OMNIS/HTML folder. The Submit task can be inherited from the monitor task.

The first pane in the wizard lets you define the fields to appear in your html Submit form. A Submit and Reset button are included on the form automatically.


OMNIS Remote Task Wizard - Submit
Select the controls you require on your HTML form
Control Type
Name
Properties
Add
Update
< Previous
Next >
Cancel

Control Type: Text Field
Name: TextField1
Label: TextField
[✓] Inherit from Monitor task

You will need to amend the $construct() code of the remote task to return the URL you want to return when the user has pressed the Submit button on the form. This is documented in the task's $construct() method.

Creating a New Remote Task Class

Normally you would use the wizards and templates in the Component Store to create a remote task, but you can create a completely empty task using the New Remote Task template either from the Component Store or using the Class>>New option in the Class Browser.

To create a new remote task class

- Open your library in the OMNIS Class Browser
- Display the classes in your library using the View>>Down One Level menu option on the Browser menu bar
- Drag the template called "New Remote Task" from the Component Store onto the Browser

or

- Select Class>>New>>Remote Task from the Browser menu bar
- Name the new task
- Double-click on the task class to modify it

You modify a remote task class in the method editor. You can place in the $construct() method any code that you want to run when the remote task is instantiated. You can add any other custom properties and methods to the task, as well as any type of variable including instance variables.

Remote Task Instances

When the OMNIS Web Client first connects, OMNIS first creates an instance of the Remote Task Class as specified by the Remote form class to which you are connecting ($designtask property of the form). The $construct method of the Remote Task Class can specify a parameter variable of type Row as its first parameter, containing up to 9 optional parameters that you can specify in your html code. If you want to check the value of the Param1 parameter you can use *params.pOption1* assuming you named your row variable *params*.

When the remote task $construct method terminates it should do so without returning a value if the connection is to go ahead, that is, *Quit method* without specifying a return value. If you need to return an error you can use *Quit method 'The Error message'*. In this case the task instance is destroyed and the connection is severed. If you don't return an error, OMNIS will now instantiate an instance of the remote form class and call the $construct method of the form.

Events

For remote tasks, the **evBusy** and **evIdle** events are sent to the $event() method during the lifetime of a connection: evBusy is sent when OMNIS receives a request from a client, evIdle is sent when OMNIS is about to return the result of a request. The following example, shows the code for the $event() method in the Monitor task created using the Monitor task wizard:

```
On evBusy
  If iMonitorOpen
    Do iMonitorRef.$setstatus($cinst,kTrue) Returns lServerBusyFlag
    ; iMonitorRef is a reference to a Monitor window
    ; created by the Monitor task wizard
    If lServerBusyFlag
      Quit event handler (Discard event)
    End If
  End If
On evIdle
  If iMonitorOpen
    Do iMonitorRef.$setstatus($cinst,kFalse)
  End If
```

In addition, tasks report the **evRejected** event which is generated when OMNIS rejects a connection by a client. Usually this occurs if there are two many users trying to connect to OMNIS, or $maxusers of the remote task has been exceeded. The parameter pErrorText is "Too many users connecting to server" for the first case, and "Too many users connecting to task [taskname]" for the second.

Client Access Statistics

Remote tasks instances have a number of properties that let you monitor connections to your OMNIS server application.

– **$connectbytessent**
specifies the number of bytes which have been sent to the client during the connection. This property is set after $construct() has been executed.

– **$requests**
specifies the number of events executed on the server. Excludes connect and disconnect messages. Updated prior to evBusy message.

– **$reqtotbytesreceived**
  the total number of bytes received from the client for all requests. To calculate an average per request, you can divide this value by $requests. Updated prior to evBusy message.

– **$reqtotbytessent**
  the total number of bytes sent to the client for all requests. To calculate an average per request, you can divide this value by $requests. Updated prior to evIdle message.

– **$reqmaxbytesreceived**
  The largest block in bytes received from the client for all requests. Updated prior to evBusy message.

– **$reqmaxbytessent**
  The largest block in bytes sent to the client for all requests. Updates prior to evIdle message.

– **$reqcurbytesreceived**
  The number of bytes received from the client for the current request. Updated prior to evBusy message for the current request.

– **$reqcurbytessent**
  The number of bytes sent to the client for the current request. Updated prior to evIdle message.

Connecting using Standard HTML Forms

In addition to the OMNIS Web Client, OMNIS Web Studio lets you interact with your OMNIS application using standard html forms. In this case, your html form has to connect to an OMNIS remote task class and does not use a remote form. For example:

```
<form method="GET" action="/cgi-bin/omnisapi.dll">
<input type="hidden" name="OmnisServer" value="PortNumber">
<input type="hidden" name="OmnisClass" value="RemoteTaskName">
  <input type="hidden" name="OmnisLibrary" value="LibraryName">
<p><input type="password" name="pPassword" size="20"></p>
  <p><input type="text" name="pQuery" size="80"></p>
  <p><input type="submit" value="Submit" name="B1">
  <input type="reset" value="Reset" name="B2"></p>
</form>
```

The form has the special fields:

– **OmnisServer**
  specifies the port number or service name of the OMNIS server, that is, the value specified in the $serverport preference in the OMNIS runtime server.

– **OmnisClass**
  the name of the Remote Task Class to connect to.

– **OmnisLibrary**
  the internal name of the library to connect to, that is, the name of your library less the .lbs extension; the library and the OMNIS server must be running on the server for the form to connect.

The following example html source code implements a feedback form. The OMNIS specific parameters are marked in bold; the remainder of the source specifies the form fields and text labels in the form, including a standard Submit button.

```
<form method="GET" action="/cgi-bin/omnisapi.dll">
   <input type="hidden" name="OmnisClass" value="tThinClientFeedback">  ;;
   the remote task name
   <input type="hidden" name="OmnisLibrary" value="Webclient">  ;; the
   library name
   <input type="hidden" name="OmnisServer" value="5912">  ;; the port number
   <table border="0" cellspacing="0" cellpadding="0" width="760">
      <tr>
         <td width="788" valign="top"><div align="right">
         <p><strong><font face="Arial">Developer
               Name:</font></strong></td>
         <td width="564" height="25"><font face="Arial">
         <input type="text" name="Name" size="27"></font></td>
      </tr>
      <tr>
         <td width="788"><div align="right">
         <p><strong><font face="Arial">Serial
               No:</font></strong></td>
         <td width="564" height="25"><font face="Arial">
         <input type="text" name="Serial" size="27"></font></td>
      </tr>
      <tr>
         <td width="788" valign="top"><strong><font face="Arial">
            <div align="right">
         <p>Platform:</font></strong></td>
         <td width="564" height="23"><table border="0"
            cellspacing="0" cellpadding="0" width="520">
            <tr>
               <td width="135"><font face="Arial">
               <input type="checkbox" name="Macintosh" value="YES">
               <strong>Macintosh</strong></font></td>
               <td width="385"><font face="Arial">
               <strong><input type="checkbox" name="Windows"
                     value="YES">Windows</strong></font></td>
            </tr>
         </table>
         </td>
      </tr>
      <tr>
         <td width="788" valign="top"><strong><font face="Arial">
         <div align="right"><p>Client:</font></strong></td>
         <td width="564" height="23">
         <table border="0" cellspacing="0" cellpadding="0"
               width="601">
            <tr>
               <td width="136" height="13"><font face="Arial">
               <input type="checkbox" name="ActiveX" value="YES">
               <strong>ActiveX&nbsp;&nbsp; </strong></font></td>
               <td width="135" height="13"><font face="Arial">
               <strong><input type="checkbox" name="Netscape"
                  value="YES">Netscape </strong></font></td>
               <td width="330" height="13"><font face="Arial"><strong>
               <input type="checkbox" name="RAWHTML"
                  value="YES">RawHTML</strong></font></td>
            </tr>
         </table>
         </td>
      </tr>
      <tr>
         <td width="788" valign="top"><strong><font face="Arial">
               <div align="right">
         <p>Comments:</font></strong></td>
         <td width="564" height="249" valign="top">
         <font face="Arial"><textarea rows="11" name="Comments"
               cols="57"></textarea></font></td>
      </tr>
   </table>
   <div align="center"><center><p><font face="Arial">
   <input type="submit" value="Send Comments" name="B1"></font></p>
   </center></div>
</form>
```

When the ‘Submit’ button is pressed, the web server extension (e.g. omnisapi.dll) is executed and passed all the form’s parameters. The web server extension sends the request to the OMNIS queue. OMNIS creates an instance of the specified Remote Task Class and calls its $construct method. When the $construct terminates, the task instance is destroyed.

Task Wizards and $construct

All the remote tasks created using the remote task wizards contain a $construct() method containing a parameter variable of type Row and called pParams. All html form fields in the form can be accessed from this row variable. For example, using the form above, you can check the value of the password field using *pParams.pPassword*. Note that the column name in the row variable is the name given to the html form control.

When the $construct method of the task terminates, it can return the URL of the result page if appropriate, that is, *Quit method 'www.mywebsite.com/result.htm'*. The result could be a simple Thank You page telling the client that their details were received, or it could be an html report page generated in OMNIS using the html printing device.

If you need to return an error you can use *Quit method 'nnn The Error message'*, where *nnn* is a three digit error code followed by a single space before the error text.

Chapter 5–Deploying Your OMNIS Web Solution

Having designed your remote forms and tasks in your OMNIS library, you are ready to deploy your application to the web. You need to place your OMNIS library on a Win32 server with the OMNIS runtime, and upload your html pages to your intranet or internet web server. When users access your website for the first time, they need to download the OMNIS web client. You can use the installers supplied on the OMNIS Studio CD or create your own installers, but in all cases, users need to download the web client before using your web solution.

This chapter describes how you:

- set up your html pages containing the OMNIS Web Client;
  this involves editing the template html pages created for you by the form wizards
- install your web server and the web server extension supplied by OMNIS
- install the OMNIS runtime server
- copy your html files and web client installers to your web server;
  you may need to create your own installer for users to download from your website

Editing your Html Pages

The OMNIS Web Client is typically placed inside an html page using any of the html editors. If no html editor tool is available, the web client can be entered using a standard text editor. When you use the Remote Form Wizards to create your remote forms, OMNIS creates a template html file automatically, and places it in the **html** folder under the main OMNIS folder. You can use these templates as a basis for your html pages or copy the ActiveX and its parameters to your own html pages. Note that the templates contain the html code for the ActiveX for Internet Explorer and the Netscape plug-in, so if you are using IE only you can comment out the Netscape plug-in from the template html page. Netscape on the other hand ignores the ActiveX code. Alternatively you can use ASP code to examine the browser in use and show the appropriate web client component, either the ActiveX or Netscape plug-in depending on the browser in use on the client machine.

Embedding the ActiveX Web Client

The html code and parameters for the ActiveX are as follows.

```
<object classid=" clsid:13510606-30FA-11D2-B383-444553540000"
   width="500" height="250">
<param name="OmnisServer" value="ServerPortNumber">
<param name="OmnisLibrary" value="LibraryName">
<param name="OmnisClass" value="RemoteFormName">
<param name="WebServerUrl" value="www.yourwebserver.com">
<param name="WebServerScript" value="/cgi-bin/omnisapi.dll">
<param name="Param1" value="pOption1=Data">
  ...
<param name="Param9" value="pOption9=Data">
</object>
```

The OMNIS Web Client ActiveX has the following parameters.

– **classid**
  the id of the ActiveX object; includes the height and width which is the same as the remote form.

– **OmnisServer**
  the port number of the server, or service name, which is registered on the server, and specified in the $serverport preference in OMNIS. The OmnisServer parameter can also be in the form IpAddress:port, if the OMNIS server is running on a different machine to your web server. For example,
  “111.222.333.444:5912 “
  “111.222.333.444:omnis” ;; where OMNIS is a service name
  “www.myhost.com:5912”
  “www.myhost.com:omnis”

- **OmnisLibrary**
  the internal name of the library to connect to; the default is the library file name minus the .lbs extension.
- **OmnisClass**
  the name of the remote form to connect to.
- **WebServerUrl**
  the URL of your web server.
- **WebServerScript**
  the location of the web server extension, such as the omnisapi.dll, on your web server; typically the /cgi-bin folder.
- **Param1 to Param9**
  specifies additional parameters, which can be sent to the OMNIS server. In the value clause you have to specify the parameter name and the data separated by an equal sign.

The following html code allows a connection to a remote form called **rfBooks** in the Books example on a web server in Mitford House, OMNIS Software's main development center.

```
<object classid="clsid:13510606-30FA-11D2-B383-444553540000"
   width="298" height="287">
  <param name="_Version" value="65536">
  <param name="_ExtentX" value="7161">
  <param name="_ExtentY" value="7373">
  <param name="_StockProps" value="0">
  <param name="OmnisServer" value="5912">  ;; the port number
  <param name="OmnisLibrary" value="WEBCLIENT"> ;; the library name
  <param name="OmnisClass" value="rfBooks"> ;; the remmote form
  <param name="WebServerUrl" value="mhthinserver.mh.omnis-
   software.com">   ;; the web server name
  <param name="WebServerScript" value="/cgi-bin/omnisapi.dll">
           ;; the location of the web server extension omnisapi.dll
</object>
```

You can examine the template html pages in the OMNIS\HTML folder to see how the OMNIS Web Client is embedded. The template pages have the port number/name, library name, and remote form class parameters filled in, but you will need to change or add the other parameters, such as WebServerUrl and WebServerScript, to allow remote access to the remote form.

Embedding the Netscape Plug-in Web Client

The html code and parameters for the Netscape plug-in are as follows.

```
<EMBED  type=application/OMNIS-RCC-plugin
        name="rcc1"
        width=500
        height=500
        OmnisServer="ServerPortNumber"
        OmnisLibrary="LibraryName"
        OmnisClass="RemoteFormName"
        WebServerUrl="www.yourdomainname.com"
        WebServerScript="/cgi-bin/omnisapi.dll">
```

– **OmnisServer**
  the port number of the server, or service name, which is registered on the server, and specified in the $serverport preference in OMNIS.

– **OmnisLibrary**
  the internal name of the library to connect to; the default is the library file name minus the .lbs extension.

– **OmnisClass**
  the name of the remote form class to connect to.

– **WebServerUrl**
  the URL of your web server DLL.

– **WebServerScript**
  the location of the web server extension, such as omnisapi.dll, on your web server; typically the /cgi-bin folder.

CGI

If your web server does not support the MS ISAPI interface you will have to use the nph-omniscgi.exe cgi program. In this case, you need to add the nph-omniscgi.exe program to your cgi-bin folder and use this name in the **WebServerScript** parameter in your HTML file(s).

OMNIS Server Configuration

The *OMNIS Server* or runtime executable is the program that runs your OMNIS application. You will need to install an OMNIS runtime server with a Web serial number on a Win32 server. Note that you cannot use a standard multi-user runtime serial number for your OMNIS server, in this case you must use a web serial number designed for OMNIS Web Client access. The server does not need to be your web server, but it must be a Win32 (NT/95/98) server. For testing and debugging you can use the OMNIS development version, but for deployment you should use an OMNIS runtime executable.

Setting the OMNIS Port Number

You need to set the port number or service name of the OMNIS runtime server. The port must be available on both the OMNIS runtime server and the web server. The number can be between 1 and 32767, but you must not use 80 or any other number used for e-mail, FTP, or other services. You should use a high number, preferably a four or more digit number, for example, 5912. The OMNIS server port number is stored in the OMNIS.CFG file in the STUDIO folder.

In the development version of OMNIS you can set the port number in the OMNIS preferences via the **Tools>>Options/Preferences** menu option. This option opens the Property Manager showing the OMNIS preferences. You can set the OMNIS server port in the **serverport** property. However, since you'll be running your OMNIS web solution using the runtime version of OMNIS you have to set its server port number too. You can set the port number in the OMNIS server via the Server Configuration dialog, available under the **File>>Server Configuration** menu option (this item is visible in Win32 versions only, and only if the runtime is serialized with a web serial number). The dialog lets you enter the server port and server maximum requests preferences. If the OMNIS server is already listening, you are prompted to restart OMNIS, otherwise the OMNIS runtime server starts to listen on the specified port.

You can set the OMNIS server port using the notation, *Calculate $prefs.$serverport as 5912*, although you must ensure that the value matches whatever you have specified in your html pages containing the web client.

The web server extension (omnisapi.dll) communicates via TCP/IP with OMNIS. Therefore your web server and the server hosting your OMNIS runtime both require WinSock.

NT Servers

For NT servers only, a program called NTSERV.EXE is provided on the Studio CD, which installs an NT service to run the OMNIS server. By default the NT service name is *OMNIS Server*. Do not confuse NT service with the TCP/IP service names which you can specify in the $serverport property for communication. They are unrelated.

If you did not set up OMNIS as an NT service during installation you can register it now by running NTSERV.EXE with the following command line:

```
NTSERV -install
```

To remove OMNIS as a service, run the following from the command line:

```
NTSERV -remove
```

To run OMNIS as console application for debugging, run the following from the command line:

```
NTSERV -debug <params>
```

It is important that whenever you use the NTSERV.exe that the file is in the same folder as the OMNIS.exe that you are using.

Web Server Configuration

Installing the Web Server Extension

You need to install the web server extension OMNISAPI.DLL, supplied by OMNIS, on your web server inside your /cgi-bin folder of your web site. To connect to the web server extension, the OMNIS runtime and your OMNIS library have to be running on the server. The number of possible concurrent connections is specified by the OMNIS runtime web serial number.

Typically a web server will have a place to store all executable code accessible via HTTP over the Internet. This is often the /cgi-bin folder, but it need not be. You need to place the web server extension, such as the omnisapi.dll, in this folder and ensure that the web server is set up correctly to enable it to be executed. Finally, you will need to add the omnisapi.dll path as the WebServerScript parameter in your html files, for example WebServerScript=/cgi-bin/omnisapi.dll.

If your web server does not support the MS ISAPI interface you will have to use the nph-omniscgi.exe cgi program and edit your html pages accordingly.

ISP Web Hosting

If your website is hosted by a third-party, typically an ISP, they will need to place the OMNIS web server extension in their cgi-bin folder, and furthermore they need to provide you with a direct connection to the Internet. Your ISP may want to test the web server extension, usually the case for any files you place in their cgi-bin folder, and this is often very expensive. Alternatively, you may consider having your own “local” web server specifically for running your OMNIS web solution and, if necessary, link to it from your main website hosted by your ISP.

If your OMNIS web solution uses a website hosted by an ISP you will need to adjust your port settings in the OMNIS server and html files. In this case you can use DomainName:Port or IPAddress:Port in your port setting.

Secure Sockets

You can use secure sockets, if available, if either the page containing the OMNIS Web Client is secure, or your WebServerUrl specifies a full URL using the https: prefix as follows:

https://remainderOfFullURL.

OMNIS Web Client Installers

To use your OMNIS web solution, a user must first download and install the OMNIS Web Client. You can provide a link on your website to allow the user to download an installer. Installers for both the ActiveX or Netscape plug-ins are provided on the OMNIS Studio CD, and they are freely available on the OMNIS website via the Home page at http://www.omnis-software.com.

Creating your own Installers

In most cases, you can use the OMNIS Web Client installers we provide. For example, you can link to the OMNIS website Home page to allow users to get the latest version of the web client, or you can place the installers on your website with the appropriate links. Alternatively, you may want to create your own web client installer, either containing a subset of the components we provide as standard, or if you want to add your own web components. You can use any industry-standard Installer package, such as InstallShield, to create your own web client installers. This section describes the web components that comprise the OMNIS Web Client.

Web Client Components

As explained previously, the OMNIS Web Client is an ActiveX control or Netscape plug in. These are small objects, which dynamically link in further DLLs (Shared Libraries on the Macintosh). You may want to provide an installer for both the ActiveX and the Netscape plug-in, depending on which browser your html pages/application supports. The following components make up the OMNIS Web Client.

– **Orfc.ocx** & **Orfc.dat**
  the ActiveX object
  or
  np_orfc.dll & **np_orfc.dat**
  for the Netscape plug-in

– **Orfcgui.dll**
  library that provides the web client GUI (required)

– **Orfcmain.dll**
  the web client engine (required)

Remote Form Components

The remote form class objects are external components written specially for the remote form class. No other OMNIS objects can be placed in a remote form class, except the built-in Page pane control. The web components are contained in the OMNIS Web Client installers that we provide on the Studio CD and available to download from our website, so in most cases you can use these. If you are building your own installers you need to include the web components, although you may want to omit any components that are not used in your application.

The form components are shipped in the following libraries, each containing a number of remote form class components:

– **Formback.dll**
  contains the background Label object and Border web components.

– **Formflds.dll**
  contains the Push button, Single line entry field, Multi line entry field, Checkbox, Radio group, List box, Drop list, Combo box, Heading list and Picture web components.

– **Formsbar.dll**
  contains the Sidebar control.

– **Formtbar.dll**
  contains the Tab bar control.

Chapter 6– Troubleshooting Guide and FAQ

This chapter contains notes and FAQs that hopefully solve some of the problems you may be having regarding the OMNIS Web Client.

General

Q: I have an existing OMNIS Studio application. Do I need to start from scratch to take advantage of the OMNIS Web Client, or can I adapt my existing application?
A: Yes, you can adapt your existing application to use the OMNIS Web Client to give parts of your application a web interface. Developing an OMNIS web solution using the OMNIS Web Client is exactly the same as designing any other OMNIS application. In addition, if you have an OMNIS 7 application, you can convert it to Studio and adapt it for web access using the web client.

Remote Forms

Q: I've added an event handling method behind a button, but when I test the form and click on the button, nothing happens, the method does not appear to run. Why is this?
A: It's probably because you haven't enabled the evClick event for the pushbutton control. By default, all events for all form controls are disabled; you have to enable individual events for an object in its **events** property.

Q: Should I worry about my Windows users using small or large fonts?
A: No, remote form objects use TrueType fonts only, and the client browser scales the font size at runtime depending on what the client machine has set.

Q: I want a particular field to get the focus when my remote is opened. How can I do this?
A: Specify the number of the field, set in the **order** property of the field, in the **startfield** property of the remote form.

Browser

Q: I have designed my remote form, but when I press Ctrl/Cmnd-T my web browser does not appear and I don’t get to see the form. Why is this?
A: Have you installed the web client? You must install the OMNIS Web Client to design and test your remote forms while using the development version of OMNIS. If the web client is not installed and you hit Ctrl/Cmnd-T to test the form, your web browser will not appear. To fix this, install the web client using one of the installers on the Studio CD.
If you have installed the web client and still don’t get to see the form, this may be because the ActiveX is not registered properly. The installer registers the ActiveX automatically, but if this has failed, for some unknown reason, you can do it manually using the REGSVR32.EXE program. Under Windows, open the Run dialog from the Start button, and enter the following command, using the correct path to the .OCX:

```
regsvr32.exe c:\windows\Webclient\Orfc.ocx
```

Q: Ctrl/Cmnd-T shows two forms with Internet Explorer. Why is this?
A: If you have Netscape Navigator on your machine as well as Internet Explorer (IE) and you have installed both the ActiveX and Netscape plug-in, IE may try to use the Netscape plug-in controls from Navigator and display both objects. Ctrl/Cmnd-T uses a template file in the HTML folder to create a test HTML page. This file has the html source for both the Netscape and Explorer objects to allow testing for both browsers. To stop this, try removing the source for either the ActiveX or Netscape object from the **template.htm** file, deleting any HTML files already created by Ctrl-T in the HTML folder (do not delete template.htm), and then retest using Ctrl/Cmnd-T.

Q: Why do I get the “unable to locate class” message?
A: Either the name of your library and the OmnisLibrary parameter in your HMTL file do not match, or your library is not open; the OMNIS runtime server and your library must be running at all times to allow remote access from a browser.

Q: Why do I keep getting the IE Security Alert message when hitting Ctrl/Cmnd-T?
A: Maybe you’re using an old beta version of the OMNIS Web Client. In this case you should upgrade to the latest version of the client which has no security alerts.

OMNIS Server

Q: Do I set the OMNIS serverport property to 80?
A: No! OMNIS should not be set to port 80, or any other number or service name already in use. OMNIS needs the service of a standard web server. Therefore set the serverport in OMNIS to a relatively high number, containing at least four digits, although you may need your MIS dept to allocate a number. And remember, the server port number in OMNIS should be the same as specified in the OmnisServer parameter in your HTML file(s).

Q: Can I serialize the OMNIS runtime running my web solution, using my existing multi-user serial number?
A: No. You must serialize the OMNIS runtime server using a special web serial number. These numbers contain a “W” and allow up to a specified number of concurrent users. Contact your local sales representative to purchase a serial number.

Q: Why do I get a “server busy” message?
A: There’s too many users, the server is already full. You may also get this message on a development version if the browser connection is not closed correctly. In this case, OMNIS thinks users are still connected. Such connections will eventually timeout, but to get around this, you can close and re-open your library.

Chapter 7–Reference

This chapter lists the events and notation for the OMNIS Web Client, including the notation for remote forms and remote tasks.

Events

Remote Task Classes

Event	**Description**
evBusy	sent when OMNIS has received a request from a client
evIdle	sent when OMNIS is about to return the result of the request
evRejected	generated when OMNIS rejects a connection by a client; usually this occurs if there are two many users trying to connect to OMNIS, or $maxusers of the task has been exceeded.

Remote Form Objects

All remote form objects send the evBefore and evAfter events if enabled in their $events property. In addition, objects such as the pushbutton and most list types send the evClick and evDoubleClick events. Some objects have their own specific properties, such as the sidebar.

General

Events	**Description**
evBefore	sent when the user is about to *enter* the form field
evAfter	sent when the user is about to *leave* the form field
evClick	sent when the user has *clicked* on the form field
evDoubleClick	sent when the user has *double-clicked* on the form field

Sidebar

Events	Description
evIconPicked	sent when an icon in the sidebar is picked
evSetPicked	sent when an icon strip or group bar is picked
evSetBeingPicked	sent when an icon in the sidebar is being picked

Notation

OMNIS Preferences

You can set the OMNIS preferences using the Tools>>Options/Preferences option on the main OMNIS menubar, or using the notation under the $root.$prefs group.

Properties

Property	Description
$serverport	specifies the server port on which OMNIS listens for connections; a number from 1 to 32767, or a service name; this should not be set to 80 or any other number or service name already in use. The serverport setting in OMNIS should match the number or name specified in the OmnisServer parameter in your html files.
$servermaxrquests	the maximum number of concurrently executing requests to the OMNIS server; a number from 1 to 511.
$webbrowser	the name and path of the web browser that can be used for testing remote forms on your local machine; leave blank to use the default browser. Also specifies the browser used for the OMNIS Help system if no browser ActiveX object is available.

Methods

Method	Description
$clearcachedforms()	clears any remote forms cached on the OMNIS server
$promptforicon()	opens the Select an Icon dialog; implemented for Web Client templates and wizards: for development use only, you should not use this method at runtime in a deployed web solution.

Remote Form Class

Properties

Remote forms have the standard properties together with the following.

Properties	**Description**
$designtaskname	the remote task name; you must specify a remote task class name. Remote forms cannot function without a remote task class.
$iconpages	specifies a list of icon pages required for the class, these are sent to the client together with the class data. You can add icon pages using $iconpages.$add(iconID). OMNIS will add the page of the icon to $iconpages if it isn't already specified. OMNIS does not send individual icons to the client, but will always send complete pages. Therefore you should choose icons from as few pages as possible, or put all the icons for a form on one page. Individual icons pages are separated by commas, and pages from either the #ICONS system table, OMNISPIC, or USERPIC icon data files (in that order) are further separated by colons. For example, an iconpages setting of: "Books,General;;Embedded Colors" specifies two pages from the #ICONS table in the current library and one page from the USERPIC data file.

Methods

Remote form classes have the standard class methods $construct, $destruct, and $redraw together with the following. Note that $open() is for development only and should not be used at runtime.

Methods	**Description**
$open()	Design environment only: not supported in runtimes. $open() lets you test your forms while debugging your library; does the same as Ctrl/Cmnd-T keypress. When executed, OMNIS opens the html page associated with the remote form in the local browser. The html page must be located in the **html** directory in OMNIS root, and have the same name as the remote form followed by .html. When creating remote forms using the provided wizards, the html pages are created for you. You can specify an alternative html page in parameter 1, in which case the full path and file name must be provided.
$canclose()	always return kFalse. You cannot close a remote form directly. You must close its remote task.
$senddata()	$senddata(iVar1,iVar2,..) specifies the instance variables to be returned to the client at the end of executing an event on the server.
$setcurfield()	$setcurfield(FieldNumber|Name|Ident) sets the current field on the client machine. You can specify the object name, number, or ident of the field.
$showmessage()	$showmessage('Message','Title') displays an Ok message on the client machine using the specified Message and Title. Only one message can be shown for each communication. Executing $showmessage() a second time during the same communication, will replace the current message text.
$showurl()	$showurl('URL','FrameName') displays the specified html page in a new browser window or frame on the client machine. First parameter specifies the URL of the html page. The second parameter specifies the frame name. If it is empty the page is displayed in a new window, otherwise it is displayed in the specified frame of the current window.

Remote Form Objects

General Properties

Remote form objects have the standard properties of a window class object together with the following. Note all remote form objects are external components.

Property	Description
$componentlib	the component library name; one of the following: Formback, Formflds, Formsbar, or Formtbar
$componentctrl	the control type
$objtype	external component of type kComponent
$events	specifies the events enabled for the object; only those events that are enabled are sent to the server
$dataname	Must only specify instance variables

Combo Box Object

Property	Description
$dataname	dataname of the entry field part of the combo box
$listname	the list variable for the list part of the combo box

Radio Group Object

Property	Description
$horizontal	if true, radio buttons are drawn in horizontal columns, left to right.
$columncount	specifies the number of columns to divide the specified number of buttons into (default is one). 9 Radio buttons and 3 columns would display a 3x3 grid of buttons. The width of each column is determined by the longest piece of text.
$minvalue	integer value of the first radio button (default is zero)
$maxvalue	integer value of the last radio button (default is two). Together, the $minvalue and $maxvalue properties determine the number of radio buttons.

Heading List Object

Property	Description
$colcount	number of columns to be displayed.
$columnnames	comma separated list of column names.
$columnwidths	comma separated list of column widths in pixels.

Sidebar Object

Property	**Description**
$fillcolor	fill color for sidebar background
$flipswitch	controls whether or not icon bars slide when clicked; kFalse by default which means they slide
$labelcolor	icon label color
$selectedlabelcolor	selected icon label color
$selectcurrent	if true the current icon is hilited
$currenticon	specifies the current selected icon in the current group
$currentset	specifies the current selected group
$show3dundermouse	if true icons are drawn 3d if under the mouse
$showiconnames	if true the icon names are shown
$buttonfillcolor	the button fill color
$washstartcolor	the start color for sidebar wash; has effect if washstrip is enabled
$washendcolor	the end color for sidebar wash; has effect if washstrip is enabled
$washdirection	the direction for sidebar wash; has effect if washstrip is enabled
$tilebmp	the icon for a tiled background; has effect if tilestrip is enabled
$tilestrip	if true the sidebar background is tiled
$washstrip	if true the sidebar background has a wash

Tab bar Object

Property	**Description**
$nosofttab	the number of tabs
$currenttab	the current tab
$tabtext	the text for the current tab
$tabtip	the tip text for the current tab
$position	the position or orientation of the tabs, either Top, Bottom, Left, or Right
$style	the style of the tabs; a constant: kDefaultWebTab, kSquareWebTab, kTriangleWebTab

Methods

Remote form objects contain the $redraw() method, which you can execute using the notation ObjectName.$redraw() or $cobj.$redraw().

Remote Task Class

Properties

Property	Description
$maxtime	the maximum amount of time, in minutes, the user is allowed to stay connected; if set to zero there is no limit.
$timeout	the amount of time (in minutes), which can elapse, while idle; if set to zero there is no timeout.
$maxusers	the maximum number of clients which can simultaneously connect to this task; if zero, there is no limit.

Methods

Methods	Description
$construct()	called when a client connects to the task. The first parameter contains a row variable. This row variable contains the connection parameters.
$destruct()	called when the client disconnects.
$open()	Design environment only: not supported in runtimes. $open lets you test remote tasks while debugging your library. When executed, OMNIS will open the html page associated with the remote task in the local browser. The html page must be located in the **html** folder in OMNIS root, and have the same name as the remote task followed by .html. When creating remote tasks using the provided wizards, the html pages are created for you. You can specify an alternative html page in parameter 1, in which case the full path and file name must be provided.
$close()	Executing $close forces the OMNIS server to disconnect the client and close all form instances, before closing the task instance.
$event()	handles the events for the task instance: the following events are sent during the lifetime of a connection via the OMNIS Web Client. evBusy - sent when OMNIS has received a request from the client. evIdle - sent when OMNIS is about to return the result of the request. evRejected - sent when OMNIS rejects a connection by a client. Note: When receiving the evBusy event, you can prevent any processing by discarding the event. In this case the client receives a “Server busy” message.

Remote Task Instance

Properties

Property	Description
$ident	Unique ID in range 1 to 2^31. It is reset when OMNIS shuts down or the last client disconnects
$order	The index value of the task instance. (Only unique for instances of the same task class, each task class maintains its own array of instances)
$maxtime	The maximum amount of time (in minutes), measured from $connectiontime, the user is allowed to stay connected.
$timeout	The amount of time (in minutes), measured from $lastresponse, which can elapse, before the connection is severed.
$clientaddress	The TCP/IP address of the client
$connectiontime	The date and time the user connected
$lastresponse	The date and time the user has last communicated with the server
$connectbytessent	Specifies the number of bytes which have been sent to the client during the connect. This property is set after $construct has been executed.
$requests	Specifies the number of events executed on the server. Excludes connect and disconnect messages. Updated prior to evBusy message.
$reqtotbytesreceived	The total number of bytes received from the client for all requests. To calculate an average per request, you can divide this value by $requests. Updated prior to evBusy message.
$reqtotbytessent	The total number of bytes sent to the client for all requests. To calculate an average per request, you can divide this value by $requests. Updated prior to evIdle message.
$reqmaxbytesreceived	The largest block in bytes received from the client for all requests. Updated prior to evBusy message.
$reqmaxbytessent	The largest block in bytes sent to the client for all requests. Updates prior to evIdle message.
$reqcurbytesreceived	The number of bytes received from the client for the current request. Updated prior to evBusy message for the current request.
$reqcurbytessent	The number of bytes sent to the client for the current request. Updated prior to evIdle message.

Index

#ICONS, 61, 70

$buttonfillcolor, 101
$canclose(), 99
$clearcachedforms(), 69, 97
$clientaddress, 103
$close(), 102
$colcount, 100
$columncount, 100
$columnnames, 100
$columnwidths, 100
$componentctrl, 100
$componentlib, 100
$connectbytessent, 103
$connectiontime, 103
$construct(), 78, 102
$construct, Remote tasks, 83
$currenticon, 101
$currentset, 101
$currenttab, 101
$dataname, 100
$designtaskname, 98
$destruct(), 102
$event(), 33, 70, 102
    tasks, 79
$events, 70, 96, 100
$fillcolor, 101
$flipswitch, 101
$horizontal, 100
$iconpages, 98
$ident, 103
$labelcolor, 101
$lastresponse, 103
$listname, 100
$maxtime, 102, 103
$maxusers, 79, 102
$maxvalue, 100
$minvalue, 100
$nosofttab, 101
$objtype, 100
$open(), 99, 102
$order, 103
$position, 101
$promptforicon(), 97
$reqcurbytesreceived, 103
$reqcurbytessent, 103
$reqmaxbytesreceived, 103
$reqmaxbytessent, 103
$reqtotbytesreceived, 103
$reqtotbytessent, 103
$requests, 103
$selectcurrent, 101
$selectedlabelcolor, 101
$senddata(), 67, 99
$servermaxrquests, 97
$serverport, 97
$setcurfield(), 99
$show3dundermouse, 101
$showiconnames, 101
$showmessage(), 68, 99
$showurl(), 99
$style, 101
$tabtext, 101
$tabtip, 101
$tilebmp, 101
$tilestrip, 101
$timeout, 102, 103
$washdirection, 101
$washendcolor, 101
$washstartcolor, 101
$washstrip, 101
$webbrowser, 97

80, port setting, 88
80, server port, 94

ActiveX, 9, 85
    Html, 85
Alert message, 94
Amend Startup Task option, 75

Border Object, Form control, 63
Breakpoints, 36
Browser, 9
Buttons, 63

Caching, 69
CGI, 11

    nph-omniscgi.exe, 87
cgi-bin folder, 89
    web server, 11
Check box, Form control, 64
classid, 85
Client, 8, 9
    Web Client, 9
Combo box, Form control, 65
Combo boxes
    Properties, 100
Component Store, 62
    Creating form controls, 62
    Creating remote task classes, 73, 74
    Creating task classes, 78
Components
    Form controls, 12
    Remote forms, 91
    Web client, 91
Components, Remote form, 62
Connections pane, Monitor, 75
Controls, Remote form, 62
Create New Monitor Task option, 74
Ctrl/Cmnd-T, 10, 25
currentpage property, 66

Data handling
    Optimizing, 67
Debugging, 36
Deploying remote forms, 37
Deploying your OMNIS web solution, 84
displayformat property, 64
Drop list, Form control, 65

Entry fields, 63
evAfter, 70, 96
evBefore, 70, 96
evBusy, 79, 96
evClick, 70, 96
evDoubleClick, 70, 96
Event handling methods, 33
Events, 10, 33, 70, 96
    Enabling, 93
    General, 96
    Remote Form Objects, 96
    Remote Task Classes, 96
    Remote tasks, 79
    Sidebar, 97
events property, 93
evIconPicked, 33, 70, 97
evIdle, 79, 80
evRejected, 79, 96
evSetBeingPicked, 70, 97
evSetPicked, 70, 97
Extension
    Web server, 11
External components
    Form controls, 12

FAQs, Web Client, 93
Fields, 63
    Display format, 64
Form caching, 69
Form fields, 27
Formback.dll, 92
Formflds.dll, 92
Formsbar.dll, 92
Formtbar.dll, 92

Headed list, Form control, 65
Heading lists
    Properties, 100
History pane, Monitor, 75
HTML
    ActiveX, 85
    Netscape, 87
Html files, 38
HTML forms, 12, 80
HTML Report task wizard, 76
HTTP, 9
HTTPS, 9

Icon pages, 61, 68, 70
iconpages property, 61
Icons, 19
Inherit from Monitor task option, 74
Installers, 91
    Creating your own, 91
Internet Explorer, 85, 94
ISAPI, 11
ISP web hosting, 89

Label, Form control, 63
List box, Form control, 64
Lists, 64

Messages, 68
Methods, 28, 33
    Remote Form Class, 99

Remote Task Class, 102
Monitor task wizard, 74
Monitor window, 75
Multiform Form Wizard, 58
Multi-line entry Form control, 64

Netscape, 85
Netscape Navigator, 94
Netscape plug-in, 9
Html, 87
Notation, 97
np_orfc.dat, 91
np_orfc.dll, 91
nph-omniscgi.exe, 11
NT Servers, 89
NTSERV.EXE, 89

OK message command, 68
OMNIS Hub, 11
OMNIS Library, 12
OMNIS preferences, 97
Server port, 88
OMNIS Runtime, 11
OMNIS server, 11, 37, 88
Port number, 88
Server usage, Monitor, 75
OMNIS Web Client, 9
OMNIS web server extension
Installation, 89
OMNIS.CFG
Port setting, 88
OMNISAPI.DLL, 11, 89
OmnisClass parameter, 39, 86, 87
for direct access, 81
OmnisLibrary parameter, 39, 86, 87
for direct access, 81
OMNISPIC, 61, 70
OmnisServer parameter, 39, 85, 87, 94
for direct access, 81
Open window command, 68
Open windows, 68
order property, 93
Orfc.dat, 91
Orfc.ocx, 91
Orfcgui.dll, 91
Orfcmain.dll, 91

Page pane, 91
Page pane, Form control, 66
pagecount property, 66
Param1 to Param9 parameters, 86
Password Form Wizard, 52
Picture fields, 34
Picture, Form control, 65
Plain Form Wizard, 43
Plain task wizard, 73
Port number, 88
pParams, Remote tasks, 83
Prompt for data file, 37
Prompt for data file command, 68
Prompt for input command, 68
Prompts, 68
Properties
Remote Form Class, 98
Remote Form Objects, 100
Remote Task Class, 102
Remote Task Instance, 103
Pushbutton, Form control, 63
Pushbuttons, 93

Radio Group, Form control, 64
Radio groups
Properties, 100
Reference, 96
Registering the Web Client, 10
Remote form classes
Properties and methods, 98
Remote form Components, 91
Remote form instances, 69
Remote form objects
Events, 70
Properties and methods, 100
Remote forms, 8, 41
Caching, 69
Controls, 62
Creating new, 61
Debugging, 61
Deploying, 37
Icons, 61
Programming, 67
Properties, 61
Testing, 36
Tutorial, 14
Wizards, 41
Remote task classes, 72
Creating new, 78
Events, 96
Properties and methods, 102

Remote task instances
  Properties and methods, 103
Remote task wizards, 83
  HTML Report, 76
  Monitor, 74
  Plain, 73
  Submit, 77
Remote tasks
  $construct, 83
  Events, 79
  Instances, 78
  Statistics, 79
Runtime, 11

Secure sockets, 90
Select an Icon dialog, 19
Serial numbers, 95
Server, 8, 10
  Installation, 88
Server busy message, 95
Server Configuration dialog, 88
Server port, Setting the OMNIS, 94
Server Usage pane, Monitor, 75
serverport property, 88
Servers
  NT service, 89
Sidebar
  Events, 97
Sidebar Form Wizard, 15, 54
Sidebar with pages Form Wizard, 57
Sidebar, Form control, 65
Sidebars
  Properties, 101
Single line entry Form control, 63
Smart lists, 69
startfield property, 93
Submit Form Wizard, 50
Submit task wizard, 77

Tab bar, Form control, 66
Tab bars
  Properties, 101
Tabs Form Wizard, 44
Task statistics, 79
Tasks
  Remote tasks, 72
Templates
  template.htm file, 94
Troubleshooting, 93
Tutorial, 13
  Remote forms, 14

Use Existing Monitor Task option, 74
USERPIC, 61, 70

Variables, 28

Web Browser, 9
Web Client, 9
  Installation, 94
  Installers, 91
  Installing, 10
  Registering, 10
Web Client Components, 91
Web components, 62
Web server, 10, 38
  Configuration, 89
Web server extension, 11
  Installation, 89
WebServerScript parameter, 39, 86, 87
WebServerURL parameter, 39, 86, 87
Wizard Form Wizard, 49
Working message command, 68

Yes/No message command, 68