

Specifies the command line arguments for the executable target. When the target is Run, Optima++ will pass the arguments you specify here to the program.

Allows you to stop the program either at startup or before any code is executed.

Causes this target to execute without any special breakpoints. All of the targets usual breakpoints, which are specified in the Breakpoints window, will remain active.

Causes this target to break at the first line of executable code. This allows you to debug the code that is generated by Optima++.

Causes this target to break before any of the application's own code is executed. This allows you to debug the startup sequence.

Runs the program on the computer where it is being developed. You may also debug an application on another (remote) computer.

Runs the program on a remote machine. Debugging is done on the local (this) machine.

Specifies the name of the remote computer. The name can be a dotted decimal IP address (for example, 123.45.6.78) or it can be a symbolic machine name (for example, iron.network.com).

For symbolic names, the @ symbol is replaced by a period.

Forces Optima++ to connect to the socket given below on the remote machine.

Allows you to specify whether the application is run on this computer or a remote computer connected by a network.

Allows you to specify how your WebApplication target should be run. Since a WebApplication target consists of several different types of files, there are different ways that your target can be run: By running a web server, a web browser, or just by copying the web files to a specified site.

Causes Optima++ to copy the web files to a specified site when the target is run. No executable program, such as a web browser, will be run.

Copies web files to the location you specify, then launches a web browser to allow you to navigate a web site.

You can specify where the files will be copied (published) on the **Publish** page. To specify which web browser will be launched, click **Configure**.

Allows you to choose the web browser that will be launched when the application is run.

Specifies the web browser that will be launched when the WebApplicaiton is run. To change the web browser, click **Configure**.

Copies web files to the location you specify, then launches a web server.

You can specify where the files will be copied (published) on the **Publish** page. To specify which web server is used, click **Configure**.

Specifies the web server that will be used to debug this WebApplications. To change the web server, click **Configure**.

Allows you to specify what web server will be used when this WebApplication is run.

Allows you to specify a location where web files will be copied when this WebApplication is run. Copying web files to a web site is called publishing because they are used for debugging or production.

Allows you to specify the name or ID of the running process to which Optima++ will connect in order to debug this WebApplication. If you leave this field empty, you can select a running process from a list when you click Run.

Allows you to select a process from a list of running processes.

Causes Optima++ to restart the NT service representing your web server when the WebApplication is run.

Allows you to specify the web browser that will be launched when this WebApplication is run.

When you run a WebApplication, Optima++ publishes (copies) the web files to a location you specify, then starts a web browser. You can use the web browser to navigate through the web pages and debug your application.

Launches the Microsoft Internet Explorer when this WebApplication is run.

When you run a WebApplication, Optima++ publishes (copies) the web files to a location you specify, then starts a web browser. You can use the web browser to navigate through the web pages and debug your application.

Launches the Netscape Navigator when this WebApplication is run.

When you run a WebApplication, Optima++ publishes (copies) the web files to a location you specify, then starts a web browser. You can use the web browser to navigate through the web pages and debug your application.

Launches a web browser that you specify when this WebApplication is run.


Specifies the name and location of the web browser that you want to use to test your WebApplication. Enter the name of the browser, including its full path. To select a web browser from disk, click **Browse**.

Specifies the URL of the page that the browser will display when this WebApplication is run.

URL: Universal Resource Location. You can specify a file name such as "c:\webtest\main.html" or a web address such as "www.ngo.com/dogood.html".

Lists the currently running processes and allows you to select the process that uses your DLL. When you "run" this target, Optima++ will attach to the process you specify. You will be able to debug your DLL when it is used by the process.

Allows you to specify the name or ID of the running process that Optima++ will connect to in order to debug this target. If you leave this field empty, you can select a running process from a list when you click Run.

For more information, click 

Specifies that the process that uses your DLL is an NT service that needs to be restarted. When you "run" your DLL, Optima++ will automatically start the service that you specify.

Specifies the name of the service that needs to be started when your target is run.

Specifies the name of the application that will be launched when you "run" your DLL. You can use this application to test your DLL, since a DLL can not be run on its own.

You can specify the program's path and name, and any command line parameters that are necessary.

Allows you to select an application from disk.


Allows you to specify what will happen when you run this target.

Since this target is a DLL and cannot be executed, you can choose to run an executable file. If you can choose to launch another program or attach to a running process. Optima++ will allow you to debug your DLL as it is used by the program or process.

Disallows you from running this target under the debugger's control. If you want to debug this DLL, you should not chose this option.

Starts a program that uses this DLL when you run this target. This allows you to debug this DLL while it is being used by another program. When you run the DLL, Optima++ will start the program that you specify and run it under debugger control.


To specify the program that will be run, click **Configure**.

For more information, click 

Specifies the program that will be started when you run this DLL. To change the program that will be run, click **Configure**.


Allows you to specify what program will be started when you run this DLL.

This allows you to debug this DLL while it is being used by another program. When you run the DLL, Optima++ will start the program that you specify and run it under debugger control.

For more information, click 


Attaches to a running process that uses this DLL when you run this target. This allows you to debug this DLL while it is being used by another process. When you run the DLL, Optima++ will attach to the process.

To specify the process, click **Configure**.

For more information, click 

Specifies the process that will be used to debug this DLL. Optima++ attaches to the process that you specify, allowing you to debug your DLL as it is used by the process. To change the process that will be used, click **Configure**.

Allows you to change the process that will be used to debug this DLL. When you run the DLL, Optima++ will attach to the process that you specify and allow you to debug your DLL as it is used by the process.

For more information, click 

Allows you to specify the location to which the DLL will be copied before being run. When you run the DLL, Optima++ will copy the DLL to the location you specify, then start the program or attach to the process that that you specify.

A DLL must be accessible by programs that use it. Usually this means that the DLL must be in the path or the Windows system directory.

Leaves the DLL in the target folder when you run the target.

A DLL must be accessible by programs that use it. Usually this means that the DLL must be in the path or the Windows system directory.

Copies the DLL to the location you specify before running the target. When you run the DLL, Optima++ will copy the DLL to the location you specify, then start the program or attach to the process that that you specify.

A DLL must be accessible by programs that use it. Usually this means that the DLL must be in the path or the Windows system directory.

Specifies the location where the DLL will be copied when the target is run. When you run the DLL, Optima++ will copy the DLL to the location you specify, then start the program or attach to the process that that you specify.

A DLL must be accessible by programs that use it. Usually this means that the DLL must be in the path or the Windows system directory.

Allows you to choose a location from disk. When you run the DLL, Optima++ will copy the DLL to the location you specify, then start the program or attach to the process that that you specify.

A DLL must be accessible by programs that use it. Usually this means that the DLL must be in the path or the Windows system directory.

Specifies how source files will be displayed for editing.

Specifies how source files will be displayed while debugging.

Causes Optima++ to open a separate editor windows for each event or function that you edit. This allows you to edit smaller units of code in each window.

Causes Optima++ to use the same window for all members of a class. This allows you to see the entire class and member function definitions in one window.

Displays the code that Optima++ generates for a class in the class' editor window. You cannot edit generated code.

Prevents you from changing source code while you are debugging a program.

If you change code while debugging the changes will not take effect until you rebuild the target, and the changes can make it difficult to debug the program further. For example, breakpoints can be misplaced and incorrect lines of source code may be displayed.

Reuses the same editor window to display code at the execution location while you are debugging. In other words, when execution jumps from one source code file to another, the Optima++ will open the other source file in the same editor window as the previous file.

Uncheck this option if you want to open a separate editor window for each code file being debugged.

Turning this option on minimizes the disruption caused by jumping to another source file while debugging.

Turning this option off allows you to view several source files simultaneously while debugging.

Lists the available source control systems. Source control systems allow you to share files and projects between several developers without accidentally overwriting another developer's changes.

For more information, click [»»](#)

Displays source control operations, such as checking in a file, as they are executed.

This option allows you to see what operations are performed. If you uncheck this option Optima++ will perform source control operations without displaying individual commands.

Pauses after each source control operation, allowing you to see what operations are being performed. This option is useful if source control operations are performed too quickly for you to recognize them.

Instructs Optima++ to build your project while you work.

Building while you work makes the Run menu command respond quicker and decreases system speed slightly.

Specifies the initial target type for new Optima++ projects. In other words, if you create a new project, Optima++ will automatically create a target of this type in the new project.

You should specify the target type that you most commonly create.

Specifies the whether Optima++ will create a new project when it is started.

Causes Optima++ to create a new, untitled project when it is started. The untitled project will contain a target of the type that you specify below.

Select this option if you regularly start Optima++ with the intent of creating a new project. Below, you should specify the target type that you most commonly create.

Prevents Optima++ from creating a new project when it is started.

Select this option if you usually start Optima++ in order to work on an existing project.

Allows you to select the target type that will be automatically created when you create a new project. You should select the target type that you create most often. If the target type that you select supports displaying of forms, you can select a default form type in the Form list.

Allows you to select the form type that will be automatically added to the default target for a new project. This option is only available if the default Target type supports the displaying of forms.

Specifies the file extension for the type of files for which you want to specify a default editor. Type the file extension, or select one of the registered extensions from the list.

The characters that follow the last period in a filename are called the file extension. You can specify how Optima++ will open for editing files with a given extension. This allows you to choose what editor to use for a specific type of files.

Describes the file type being configured. This description is displayed by the Windows Explorer.

You can specify how Optima++ will open for editing files with a given extension. This allows you to choose what editor to use for a specific type of files.

Specifies how files of the above type are opened by Optima++ for editing. When you double click a file of the specified type in the Files view, Optima++ will start the editor you specify to allow you to edit the file.

You can have Optima++ use its default editor, the editor specified by the system (mimicking the behavior of the Windows Explorer), or an editor program of your choice.

Causes Optima++ to open files of the specified type with the default Optima++ editor. Unless you specify a different default editor for Optima++, the standard text editor window will be used for files of this type.

In other words, when you click on a file of this type in the Files view, Optima++ will open it with its default editor, which is normally the Optima++ text editor.

Causes Optima++ to open files of the specified type with the Optima++ text editor. In other words, when you click on a file of this type in the Files view, Optima++ will open it with its text editor.

Causes Optima++ to open files of the specified type with the default Windows editor for that file type. In other words, when you click on a file of this type in the Files view, Optima++ will start the same editor as Windows Explorer.

Causes Optima++ to open files of the specified type with the program that you specify. In other words, when you click on a file of this type in the Files view, Optima++ will start the editor that you specify here.

Adds or modifies the editor settings for current file type.

Closes this dialog and discards any changes you have made.

Lists the types of files with which you have associated a particular editor program. Associating a file type with an editor will cause Optima++ to open files of that type (determined by the file's extension) with the editor you specify. In other words, when you double click a file of the specified type in the Files view, Optima++ will start your chosen editor.

Creates a new association between a file type and an editor of your choice. This allows you to use your favorite editor for any type of file.

A file's type is determined by its extension. For example, Bitmap Images have the extension ".bmp".

Removes the selected file type from the list of specially configured file types. Removing a file type from this list will cause files of that type to be opened using the standard Optima++ editor.

Allows you to modify the editor associated with the selected file type.


Displays the editor that is associated with the selected file type, and the extension of the selected file type.

Displays the icon associated with the selected file type.

Specifies the file extension for the selected file type.

Specifies the editor that will be used for the selected file type. When you open files of the selected type, Optima++ will start the editor specified here to allow you to edit the file.

Specifies the name that represents a specific folder or relative path. When a target is being built, Optima++ will replace this **Macro** with the associated **Value** to specify the locations of files such as libraries and header files. This allows you to change the location of these files by changing the value of a macro.

For more information, click 


Specifies the folder name or relative path of a specific folder. When a target is being built, Optima++ will replace the **Macro** with this **Value** to specify the locations of files such as libraries and header files. This allows you to change the location of these files by changing the value of a macro.

For more information, click [»»](#)

Adds or changes the specified macro.


Closes this dialog and discards any changes you have made.

Lists the macros that are used to define the locations of files such as libraries and header files. Use these macros in the Property Sheets of the Targets view. These macros allow you to specify different locations for building projects on different computers.


For more information, click .

Takes the selected macro out of the list.

Creates a new macro. Macros are used to define the locations of files such as libraries and header files. Use these macros in the Property Sheets of the Targets view. These macros allow you to specify different locations for building projects on different computers.

For more information, click 

Allows you to modify the selected macro. Macros are used to define the locations of files such as libraries and header files. Use these macros in the Property Sheets of the Targets view. These macros allow you to specify different locations for building projects on different computers.

For more information, click 

Includes build macros that are defined by Optima++ itself in the list above.

Lists the folders where Optima++ will search for source files while you are debugging an application. This allows you to debug parts of the application that are not located in the project folder.

Folders are searched from top to bottom.

Allows you to select from the disk a folder containing source files.

Lists folders that contain source files that will not change frequently. If your Optima++ projects use source files from other locations that do not frequently change, there is no need for Optima++ to check the files to determine whether they need to be compiled.

Specifying that a folder is read-only can dramatically decrease the time it takes to build projects that use files in the read-only folder.

Allows you to select from disk a folder that contains files that are rarely changed.

One or more of the specified options is not valid. Choose Revert to use the previously specified options, or OK to ignore the error.

Discards the options that you specified and resets the options to their previous values.

Ignores the error and applies the current options.

Note: One or more option is invalid.

This file is for context help for dynamically generated property sheets.

MTBOX -- edit the items for list boxes and combo boxes.
items.

available on the property sheet for ListBox --

MTCOLOR -- change the foreground and background color for a control. available from the Color page of forms.

DATA COLS -- the DataColumn dialog -- available for list view and grid in the database page, Browse button.

DDXFORM -- property sheet for form FDX

DDXOBJ -- property sheet for controls with FDX

formmenu.dlg

formsize.dlg

OLEFORM.DLG

query.dlg

transact.dlg

%%winmenu.dlg

Displays the form's controls in the sequence that they are selected by pressing the **Tab** key. When the user presses tab on the form, the controls are selected in the order they appear in this list. Controls are divided into tab groups by horizontal separators.

Gathers the selected controls into a group.

You should group option buttons together in order to have them work together. When the user selects one option button in a group, other option buttons in the group are automatically turned off (the `AutoRadioButton` property must be turned on for each option button in the group).

Eliminates all grouping from the form. If you do this, all controls will be considered to be one group.

Changes the tab order of the controls according to their position on the form. Controls will be ordered from top to bottom and left to right.

Creates a tab stop for the current control. As the user presses **Tab** repeatedly, controls with tab stops on the form will receive the focus in the same order that they appear in the list.

Removes the tab stop for the current control. As the user presses **Tab** repeatedly, controls with tab stops on the form will receive the focus in the same order that they appear in the list.

Moves the selected control up by one position in the list. Controls are selected in the order they appear in the list.

Moves the selected control down by one position in the list. Controls are selected in the order they appear in the list.

Closes the Tab Editor and saves any changes you have made.

Saves all the changes you have made without closing the Tab Editor.

Allows you to change the order and grouping of the tab stops for controls on the form.

Tab stops cause the control to be selected when the user presses the **Tab** key. Controls are selected in the order they appear in the list.

No Help topic is associated with this item.

Displays a grid of dots on the form design window, making sizing and alignment of objects on the form easier. The grid will not be displayed at run time.

Fixes the corners of objects to grid marks, making sizing and alignment of objects on the form easier.


Displays a grid of dots on the form design window to make it easier to align and size components visually.

Locks component placement so that all four corners are on a grid point. This makes it easier to align and size components visually.


Sets the horizontal and vertical distances in pixels between columns of grid dots.

Sets the horizontal distance in pixels between columns of grid dots.

Sets the vertical distance in pixels between rows of grid dots.

Displays the values from a column in a secondary (lookup) source as the elements of the list box. The column in the lookup source should contain all values that might be entered in the result set. The value in the bound column of the main data source is changed when the user picks a value from the list of possible values. For more information, click 

Changes the current row in the database when the selected item in this list box changes. This allows the user to change to a specific row in the database by selecting the item in the list that corresponds to that row. This property can only be used for a single selection list box (set on the Style page) and if **DataBindAsLookup** is false.

For more information, click 


Records important actions in the debug log. Information is only written to the debug log when you run a Debug version of the application. With Release versions, the TraceToLog property has no effect. Note that this property controls information written to the Optima++ debug log, not the database's trace log. Writing transactions to the Debug can help to determine the source of problems, such as consistency issues when the database is updated by another user but the displayed results are not updated.

Specifies the column in the database to which the component will be bound. The label's text will automatically change to the text stored in this column.

Specifies the query object that is the source of the data to be displayed by the check box. Use this if you want to bind the checkbox to a column in a database.

Stops the current build.

Specifies the preferred base address for loading the DLL. Set this to optimize load times as your DLL nears completion.

For more information, click 

Optima++ is building your Target. The progress meter may not change for a long period of time; please wait. To halt the build process, click **Cancel**.

Stipulates the options used when building a target for debug.

Performs no special optimization.

Maximizes the speed of your program. This will increase the size of your program.

If you specify **Full information** on the Debug page then the compiler will be unable to fully optimize your code.

Minimizes the size of your program. This will decrease the speed of your program.

If you specify **Full information** on the Debug page then the compiler will be unable to fully optimize your code.


Allows you to specify compiler optimizations individually. For more information on these options,

Lists the locations of header files. You can specify a path relative to the folder where the target is built using the `$$:` macro, or use build macros to specify a given folder:

`$$:..\MyFolder`

`$(MacroName)\RelativePath`


Optima++ searches folders in order from the top to the bottom of the list.

For more information on build macros, click 

Allows you to specify a folder to add to the list. You can specify a path relative to the folder where the target is built using the `$$:` macro, or use build macros to specify a given folder:

`$$:..\MyFolder`

`$(MacroName)\RelativePath`

For more information on build macros, click 

Inserts the folder that you typed into the list. The folder is inserted just before the highlighted folder. You can specify a path relative to the folder where the target is build, or use build macros to specify a given folder:

..\..\MyFolder

\$(MacroName)\RelativePath

Deletes the selected folder from the list.

Moves the selected folder down by one position in the list.

Optima++ searches folders in order from the top to the bottom of the list.

Moves the selected folder up by one position in the list.


Optima++ searches folders in order from the top to the bottom of the list.

Lists the locations of library files. You can specify a path relative to the folder where the target is build, or use build macros to specify a given folder:

`$^:..\lib`

`$(MacroName)\RelativePath\lib`

Optima++ searches folders in order from the top to the bottom of the list.


For more information on build macros, click 

Specifies the folder to add to the list. Type the path (absolute or relative) and click **Add**. You can specify a path relative to the folder where the target is build, or use build macros to specify a given folder:

..\..\MyFolder

\$(MacroName)\RelativePath

Optima++ searches folders in order from the top to the bottom of the list.

For more information on build macros, click 

Inserts the folder that you typed at the top of the list.

Deletes the selected folder from the list.

Allows you to control the linker settings.

Creates a separate "map file" for the module.

A map file tells you where the variables and functions in your program are located in memory, and can be useful when debugging.

Does not create a "map file" for the module.


A map file tells you where the variables and functions in your program are located in memory, and can be useful when debugging.

Specifies the name of the map file. You can specify a path relative to the directory where the target is built (that is, relative to the DEBUG or RELEASE folder for the target) by using the `$^:` macro. For example, the following will create a map file in the parent folder of the DEBUG or RELEASE folder:

```
$^:\foo.map
```

(DollarSign Caret Colon). You can also use build macros to specify a given folder:

```
$(MacroName)\RelativePath
```

For more information on build macros, click 

Allows you to specify the location on disk of the map file.

Forces your program symbols to have the exact pattern of upper and lower case letters.

Ignores the difference of upper and lower case letters for symbols.

Specifies the size of memory allocated for the program stack (bytes).

Performs floating point operations in software.

Your program will be slower, but it will run on computers without mathematical co-processors, such as 486-SX and some 386 computers.

Performs floating point operations with the math-coprocessor.

Your program will be faster, but will only run on computers with math-coprocessors installed.

Performs floating point operations with the main processor.

Your program will be slower, but it will run on computers without math-coprocessors, such as 486-SX and some 386 computers.

Click this if you want your program to run on a 286 or better computer.

Click this if you want your program to run on a 386 or higher computer.

Your program will not run on a 286 computer.

Click this if you want your program to run on a Pentium computer.

Your program will run on a 386 computer, but is optimized for the Pentium.

Lists your program's #define macros.


When you insert a macro in this list, the compiler defines that macro at the beginning of your source file.

Shows the libraries where external functions and classes are found for your Optima++ project. You can specify a path relative to the folder where the target is build, or use build macros to specify a given folder:

..\lib

\$(MacroName)\RelativePath\lib


Optima++ searches libraries in order from the top to the bottom of the list.

For more information on build macros, click 

Allows you to specify a library to add to the list. You can specify a path relative to the folder where the target is build, or use build macros to specify a given folder:

..\lib

\$(MacroName)\RelativePath\lib

For more information on build macros, click 

Inserts the library that you typed into the list.

Deletes the selected libraries from the list.

Allows you to select the name of a library with the mouse.

Specifies the name and relative path of the file.

Specifies the location on disk of the current file.

Specifies the size of the current source file in bytes.

Specifies the date when the source file was last saved.

Tells the current condition of the compiled member.

Up to date - No changes have been made since the last time the file was compiled.

Needs building - Changes have been made, and the file must be built before running the program.

Has Errors - Errors occurred when the file was compiled. Check the error log, correct the problems, and rebuild the target.

Building now - The file is currently being compiled.

Compiles the current file.

Causes no debugging information to be generated for the target. This makes debugging the target very difficult but it reduces the amount of disk space needed to build the target.

Includes line numbers in the compiled code for use with debugging. The value of some variables may not be available for debugging because of optimizations.

Provides all possible information for debugging. Choosing this option reduces the number of optimizations that can be performed.

Displays all warning messages. Warning messages can point to code bugs.

Suppresses display of compiler warning messages. Warning messages can point to code bugs.

Halts the build process when there are warnings. Warning messages can point to bugs in your code.

Continues the build process even when there are warnings. Warning messages can point to bugs in your code.

Check this to balance the importance of speed and size of your target (application, DLL, etc.).

Adds the macro to the list.

Changes the macro selected in the list to the values entered above.

Removes the selected macro from the list.

The name in the macro:

`#define` `name` `value`


The value in the macro:

```
#define          name          value
```


Empties the above text boxes and allows you to enter a new macro.

Lists the macros defined for this file. These macros will be defined when this file is compiled (or files of this type for default options).


Adds or removes component(s) to or from the system registry. You must register a component before using it in your Optima++ application.

For more information on adding and registering custom components, click 

This dialog helps you add or remove component(s) to or from the system registry. You must register a component before using it in your Optima++ application.

For more information on adding and registering custom components, click 

Lists the registered custom components and their locations. You can remove one of these components from the registry, or add new ones to this list. You must register a component before using it in your Optima++ application.

For more information on adding a custom component, click 

Removes the selected component from the system registry. Do this when you install a newer version of a component, for example.

Allows you to specify the programmatic ID of the control. The programmatic ID is the name that uniquely identifies the component in the system, and is stored in the system registry. For example, the programmatic ID of an Excel worksheet is: Excel.Sheet

Lists the registered automation servers. If a registered automation server does not appear in this list, type its programmatic ID in the text box above.


Displays only automation servers in the above list. Only use this option if the components you are looking for do not have registered type libraries (i.e. if they are not listed when the **Show type libraries** option is set).

Displays only registered type libraries in the above list.

Specifies the page on the Component Palette where the new component(s) will be placed. Choose an existing page from the list or type the name of a new page.

Uses the C++ source and header files generated previously by Optima++ to build the OLE component. This allows you to modify the contents of the source and header files that are used to build a component. You would do this in order to fix any conflicts that an object creates.

Only use this option if you have changed the component's source files and you want those changes to be incorporated in the component.

For more information, click 


Uses the `#define` directive instead of an `enum` to define the value of enumerated constants. If you uncheck this option, enumerated constants are placed in a C++ style `enum`.

Check this to fix the problem where two enumeration constants have the same name and value.


For more information, click [»](#)

Generates two member functions for each of the object's methods. One member function has the full parameter list that the object specifies. The other member function's parameter list is stored in a **WOleParmList**, which allows you to specify arbitrary optional parameters.

This doubles the number of functions generated, but allows you to specify optional OLE parameters, which are required by some objects.

For more information, click 

Declares *all* classes and methods; even those defined to be "hidden" by the object.

For more information, click 


Adds the text you specify to the beginning of the names of all classes and enumerations. Use this feature if two components use the same class name, or when you have two versions of the same component.

For more information, click [»](#)

Adds the name of an enumeration to the beginning of its enumeration constants. This solves the problem of two enumerations having items of the same name but different values. If two enumerations have the same name and the same value, you can check **Replace enumeration constants with macros using #define**.

For more information, click [»](#)

Specifies the location of the compiled interface (library and header file) to the component.

For more information, click 

Specifies that the source files that are generated by the OLE component wizard will not be deleted. These source files are not used once the wizard is finished, but you can save them if you want to modify them and regenerate the component(s).

Closes this dialog box.

Lists the component libraries that are on your component palette, and allows you to remove and add component libraries.

If you want to add an ActiveX component to your palette, close this dialog and choose **Add ActiveX Component(s)...** from the **Components** menu.

Removes the selected component library from the component palette. When you remove an ActiveX component, the component remains registered with the system until you un-register it.

Allows you to select a component library to be added to the component palette.

Specifies the name of the new file. This file may have any extension, but .cpp and .hpp are commonly used for source and header files.

Specifies which of the targets, if any, the file should be used in.

This dialog helps you create a new source or text file.

Adds the symbol and string that you typed to the list of strings for this project.

Changes the selected string and symbol to the string and symbol that you typed.

Empties the text boxes and allows you to enter a new string.

Specifies the symbol that represents this string in your program. You can use any combination of letters, numbers, and underscores in this symbol (no spaces). This is like a variable name in your program; it is useful to name the symbol with a descriptive name of what it represents in your program.

Specifies the string that represents this symbol in your program. Using a string resource in your program allows you to keep all strings in one place, and makes changing strings easier. You can use these strings, for example, as the text in a dialog or message box.

Takes the selected string resource out of the list of strings for this project.

Lists the string resources available in your program.

Allows you to manage version information. This field specifies the version of this particular target.

The product version field specifies the version of the entire product. This combination allows you to release updates of a specific target, or a completely new release.

Allows you to manage version information. This field specifies the version of the entire product.

The file version field specifies the version of this particular target. This combination allows you to release updates of a specific target, or a completely new release.

Specifies to target browsers that this version of the target is for debugging purposes. This does not affect your compiler options.

Specifies to target browsers that this version of the target is not a release version. This does not affect your compiler options.

Specifies a language for a language/character set pair. These pairs are stored in the resources, and can be used by the system to determine the character set to be used for displaying text.

Specifies a character set for a language/character set pair. These pairs are stored in the resources, and can be used by the system to determine the character set to be used for displaying text.

Specifies the version information for the current target. This information is used by the system when displaying the properties of the target.

To change the value of an item, click the item, enter the new value, and press **Enter**.

Specifies the language and character set used by this target. The system can use this information to display text correctly on computers that use another language.

Allows you to specify a new language/character set pair to be used by the target. If you define a language/character set pair, your application should provide support for users of that language.


This dialog will declare a new method for the current class. Type the prototype of the method, select the method's scope, and click **OK**.

Creates the new user function and closes this dialog box.

Allows you to enter the full prototype of the new user function; enter anything that you would normally put in the declaration of a function, including `#ifdef` conditions, and so on. You may not define a function here by entering the function body.

Allows you to select whether this function will be private, protected, or public.

public member functions are available to users of the class. private and protected member functions are not available to users of the class, but are available to other member functions of the class itself. In addition, protected member functions are available to member functions of classes derived from this class.

For more information, click 

Provides a description of the target type.

This wizard guides you through the creation of a new target (a Win32 executable, and a DLL are examples of targets).

Help for this topic has not been written.

Allows you to specify the name of the target. For an Win32 executable target, this is the name that will be given to the executable.

The new target and its supporting files will be stored in a folder with the same name as the target.

Allows you to select the drive (network or local) where the target and its supporting files will be stored.

Allows you to specify the name of the new folder where the target and its supporting files will be stored.

Shows the full path of the folder where the target and its supporting files will be stored.

Allows you to specify the name of the target. For an Win32 executable target, this is the name that will be given to the executable.

Displays a list of the types of targets that can be created. Select one of the target types by clicking its icon.

This wizard guides you through the process of creating a form for a target.

Allows you to choose the type of form that will be created. Select a form type and click **Next**.

Allows you to specify the name of the file that contains the form's description. This file is maintained by Optima+, and must have the wxf extension.

Allows you to specify the name used to access this form in code. This is not the title of the form.

Allows you to choose which target the new form will be associated with.

Specifies the ODBC data source (database) that the form will be connected to.

Specifies the user name used to connect to the database. If you do not specify a name, you will be prompted it and a password when the connection to the database is made.

Specifies the password used to connect to the database. If you do not specify a password, you will be prompted for it when the connection to the database is made.

Specifies additional connection parameters to be added to the connection string. This string is optional, and depends on what, if any, additional information is required to establish a connection with the data source you are using.


Specifies the SQL statement used for the query. This allows you to specify the columns from the database to be used, and the criteria to be used in selecting rows from the database.

Type an SQL statement or click **Edit** to build an SQL statement graphically.

Lists the predefined layouts for controls on your form. A text box and a label are created for each column in the database. Optima++ will arrange the text boxes using the layout you select here.

Determines what kind of database form will be created; a master detail form or a single query.

Master detail views have two database queries; a master query and a detail query. The information in the detail query depends on the current row in the master query.

For more information on master detail views, click 

Allows you to specify how the controls in the detail view will be arranged on the form. Select list of predefined layouts. The image above gives you a rough idea of how the controls will be arranged.

Allows you to specify how the controls in the master view will be arranged on the form. Select list of predefined layouts. The image above gives you a rough idea of how the controls will be arranged.

Specifies the SQL statement used for the query in the detail view.

Type an SQL statement or click **Edit** to build an SQL statement graphically.

Specifies the SQL statement used for the query in the master view. In an upcoming Form Wizard page, you can select the columns that will be linked to the detail view.

Type an SQL statement or click **Edit** to build an SQL statement graphically.


Attempts to connect to the database. If the connection is successful when you test it here, it is likely to be successful at run-time. If it fails, you will need to configure the database connection.

Opens the Query editor to allow you to specify database tables and columns to be used for the new form.


Specifies the caption of the group box that will be placed around the controls associated with the detail view.

Creates a data navigator control that allows the user to move the cursor in the detail view and edit the data if the detail view allows editing.


Lists the columns in the master query and allows you to select the column to be linked to a column in the detail view. Select the column to be linked to a column in the detail view, then select the column in the detail view, then click **Add**.

For more information on master detail views, click 

Lists the columns in the detail query and allows you to select the column to be linked to a column in the master view. Select the column to be linked to a column in the detail view, then select the column in the detail view, then click **Add**.


For more information on master detail views, click 

Displays the linked columns. The detail view will display records where the linked columns match the current row in the master view.

For more information on master detail views, click 


Links the selected columns in the master and detail views.

The detail view will display records where the linked columns match the current row in the master view.

For more information on master detail views, click 

Removes the link for the selected column pair.

The detail view will display records where the linked columns match the current row in the master view.

For more information on master detail views, click 

Removes all links between columns in the master and detail views.

Allows you to select the target that will be run. Select a target from the list and click **OK**.

Allows you to locate the missing file. Optima++ will automatically update the target with the file's new location.

A file reference by the target cannot be found, probably because it has been moved, renamed, or deleted. To locate the file yourself, click **Find**. To remove the file from the target, click **Remove**. To continue loading the target without the missing file, click **Ignore**.

Continues loading the target without the missing file. Click this if the file is missing temporarily (for example, if the file is located on a network drive and the computer is not connected to the network).


Removes the reference to the missing file from the target. Click this if the file is no longer used by the target. If the file's location has changed, click **Find**.

Allows you to specify the targets upon which the current target depends.


Adds the available target(s) to the list of targets that will be built before the current target. The current target "depends on" the targets in the list. So if a change is made to any target in this list, the current target will be rebuilt.

For example, if an executable target (EXE) uses functions from a library (DLL or LIB), then the executable "depends on" the library. In this case, the library should be added to this list for the executable. The library will be built before the executable to allow the executable to use the library. If a change is made to the library, then the library will be automatically re-built before re-building the executable.

Executable targets, console applications, and libraries can all depend on a DLL or a LIB or both.

For more information, click 

Lists the targets in your project upon which the current target does not depend. If the current target depends on one of the targets in this list, click the target, then click **Add**.

For more information, click 


Lists the targets upon which the current target depends. Targets in this list will be built before the current target. If a change is made to any target in this list, the current target will be rebuilt.

For example, if an executable target (EXE) uses functions from a library (DLL or LIB), then the executable "depends on" the library. In this case, the library should be added to this list for the executable. The library will be built before the executable to allow the executable to use the library.

Executable targets, console applications, and libraries can all depend on a DLL or a LIB or both.

For more information, click [»](#)

Removes the selected target from the list. Do this if the current target does not depend on the selected target. For example, if a DLL is in this list and the current target does not use any functions from the DLL, remove it from this list.

For more information, click 

Allows you to rearrange the Toolbar by adding, moving, or removing buttons. For example, you could add a button for a tasks that you perform often.

Displays larger buttons on the Toolbar and Component Palette. Uncheck this to display smaller buttons.

Using smaller buttons will make more space available on your screen by reducing the size of the main Optima++ window.

Lists the toolbars that can be displayed. To hide a particular toolbar, click its name so that it is unchecked.

Displays a popup tool-tip that describes the function of a button or the name of a component when the mouse cursor pauses over the button on the Toolbar or Component Palette.

Checks whether your program writes to memory that has not been allocated. If an invalid write is detected, a message is written to the Debug Log.

Checks that each allocated piece of memory has been properly freed by the time the program finishes execution. If not, Optima++ creates a summary of objects that were not freed properly in the Debug Log. Setting this option can help to find memory that has been allocated and not deallocated.

Produces information about each memory allocation operation as it occurs. This option will slow down execution time.

Checks the validity of the area used for dynamic memory allocation every time a new piece of memory is allocated. To produce only a summary of the number of memory allocations and deallocations, click this option so that it is unchecked.

This is one way to detect problems of one object overwriting another in this memory area.

Checks the validity of allocated objects whenever your program invokes their methods, or accesses them in some way. If an invalid object is accessed incorrectly, a message will be sent to the Debug Log and a message box will be displayed.

To use these options, build your targets in DEBUG mode.

Checks whether WString objects are valid whenever you invoke a WString method. A WString would be invalid, for example, if your program writes directly to its internal memory (which should never be done), or deletes it and then invokes one of its methods.

To use this option, build your targets in DEBUG mode.

Checks whether objects are valid whenever your program invokes their methods. An object such as a command button would be invalid, for example, if your program deletes it and then invokes one of its methods.

To use this option, build your targets in DEBUG mode.

Checks the validity of handles used to reference objects like bitmaps and fonts. A handle would be invalid, for example, if it has been deleted or freed.

To use this option, build your targets in DEBUG mode.

Lists the files in your project do not use the default build properties. To apply the default build properties to the files in the list, click **Yes**. To maintain the custom build properties for the files in the list, click **No**.

Maintains custom build properties for the files in the list instead of applying the default build properties.

Files in the list have custom build properties. In other words, when you change the default build properties the build properties of the files in the list do not change. Click **No** to maintain the custom build properties for the files in the list.

Applies the default properties to the files in the list.

Files in the list have custom build properties. In other words, when you change the default build properties the build properties of the files in the list do not change. Click **Yes** to make the properties of the files in the list the same as the default properties.

This dialog allows you to save the current form as a template that you can reuse as a base for other forms. When you saving a form as a template, Optima++ adds the form to the list of forms in the form wizard.

Displays the icon that will be used to represent this template in the Form Wizard. To modify the icon, click **Edit**. To select a different icon from disk, click **Browse**.

Specifies the name of the form template. Use a short descriptive name. Type a longer description in the box below.

Describes what this template is and why it was created. This description will be displayed in the Form Wizard when this template is selected.

Invokes the image editor to allow you to modify the icon that represents this form in the Form Wizard.

Allows you to select from disk an icon to represent this template in the Form Wizard.

Specifies the name used by other applications to access a function or variable in this DLL.

If you are accessing this library from another target created with Optima++ you do not need to specify an alias; just use the function and global variable names as they are defined in the source code.

Specifies the internal (mangled) name of a function or variable in this DLL. This name usually contains characters not used in function names since it is an internal name.

If you are accessing this library from another target created with Optima++ you do not need to specify an alias; just use the function and global variable names as they are defined in the source code.

Lists the internal (mangled) names of functions and variables in this DLL and the names used by other applications (aliases) to access them. Aliases are sometimes needed when a DLL is used by an application that expects entry-point names to be decorated differently than Optima++ does by default.

If you are accessing this library from another target created with Optima++ you do not need to specify an alias; just use the function and global variable names as they are defined in the source code.

Changes the selected alias/name pair to the new values specified.

Deletes the selected alias/name pair from the list.

Empties the edit boxes, allowing you to type in new values.


Adds to the list the alias/name pair you type.

Specifies the class or classes from which the new class will inherit. When class A inherits from class B, class B has all the functionality of class A, and can extend or redefine class A's functionality.

For more information, click [»»](#)

Specifies the name of the class. This name is used to declare instances of the class, for example. The name can contain alpha-numeric characters and underscore characters. The name is case (UPPER or lower) sensitive, so MyClass is different than mYcLaSS.

Never define a class and a struct with the same name.

For more information, click 

Specifies the name of the file used to store managed class information, information that is internal to Optima++
This name must have the WXC extension. Type a name or accept the default name.

Specifies the target that will use this class. Pick a target from the list.

Lists the places in the program where execution will stop (breakpoints). If the breakpoint specifies advanced settings, the breakpoint will only trigger when all the conditions are satisfied.

Enables every breakpoint in the project. If a breakpoint is enabled, it will be triggered. If it is disabled, it will be ignored (and execution will continue as usual).

Disables every breakpoint in the project. If a breakpoint is disabled, it will be ignored (and execution will continue as usual). If it is enabled, it will be triggered.

Removes all the breakpoints. If you don't want to remove breakpoints, but you want to ignore them, you can disable them with the **Disable All** button, or by disabling only specific breakpoints.

Allows you to specify a line of code to be executed when the breakpoint is triggered, and optionally, to continue program execution without pausing at the breakpoint.

Allows you to specify conditions under which the breakpoint will be triggered.

Closes the breakpoint editor and saves any changes you have made.

Creates a new breakpoint.

Triggers the breakpoint only if the statement entered here evaluates to TRUE, or a non-zero number.

Skips the breakpoint 'n' times before triggering, then triggers the breakpoint every time afterwards. You specify 'n' in the edit box. Uncheck to trigger the breakpoint each time it is executed.

If you have specified a condition for the breakpoint, this count (n) is only incremented when the condition is TRUE.

Executes the statement you specify when the breakpoint triggers.

If you have specified a condition and a count ('n') for the breakpoint, the condition must be satisfied 'n' times before the breakpoint is triggered.

Continues program execution when the breakpoint triggers. The statement you entered will be executed just before the breakpoint.

Shows or hides more sophisticated debugging options.

Lists the available symbols. Select the symbol whose value or address you want to use.

Specifies the name of the symbol you want to use. Type the full or partial symbol name, or select from the list.

Allows you to select a symbol to be placed in the text box.

Lists the modules used by the current target and allows you to select one.
Select the module that contains the symbol you want to use.

Lists all symbols including symbols from imported libraries. Uncheck this to display only symbols defined by in the current target.

Closes the dialog and returns the symbol you selected.

Scrolls the assembly window to the address you specify. Type the new address or click **Symbol Lookup** to select the symbol that corresponds to the address you want to see.

You can enter an address or an expression. If you enter an expression, the expression will be evaluated as an address.

Scrolls the memory window to the address you specify. Type the new address or click **Symbol Lookup** to select the symbol that corresponds to the address you want to see.

You can enter an address or an expression. If you enter an expression, the expression will be evaluated as an address.

Changes the address of the breakpoint. Type the new address or click **Symbol Lookup** to select the symbol that corresponds to the address.

You can enter an address or an expression. If you enter an expression, the expression will be evaluated as an address.

Changes the value stored in the specified memory location. Type the new value or expression.

Changes the value of the specified variable. Type the new value or expression.

Allows you to choose a symbol representing the address of the assembly code that you want to see.

Allows you to choose a symbol representing the new address of the breakpoint.

Allows you to choose a symbol representing the address in memory that you want to see.

Closes the dialog and displays the address you specified at the top of the assembly window.

Closes the dialog and displays the address you specified at the top of the memory window.

Closes the dialog and changes the breakpoint to the address you specified.

Closes the dialog and changes the memory value to the value you specified.

Closes the dialog and changes the register's value to the value you specified.

Closes the dialog and changes the variable to the value you specified.

Closes the dialog. The assembly window will not be changed.

Closes the dialog. The memory window will not be changed.

Closes the dialog. The breakpoint will not be changed.

Closes the dialog. Memory will not be changed.

Closes the dialog. The variable will not be changed.

Select the correct symbol from the list of possible matches and click **OK**.

Closes the dialog and returns the unambiguous symbol you selected.

Closes the dialog. Does not return a symbol.

Changes the indices of the first and last array elements that are shown.

Specifies the index of the first array element that you want to see.

Specifies the index of the last array element that you want to see.

Closes the dialog and changes the display bounds of the array.

Specifies the watch or inspect expression.

Closes the dialog and changes the value of the variable to the value you specified.

Just wait. Optima++ is copying the target to the remote machine.

To stop this process, click **Cancel**.

Just wait. Optima++ is building a sorted list of symbols in your program.

Specifies which target program to run and with what parameters.

Specifies the options for memory tracing.

Cuts off communications between this computer and the remote computer.

Monitors communications between this computer and the remote computer. When the lights move from left to right, the computers are communicating.

Specifies the text that you want to Find.

Specifies the text that will replace the text in the **Find What** box.

Allows you to specify the direction of the search (whether text is searched for before or after the search start position).

Searches for occurrences that are words, and not a part of a larger word.

Searches for text before the search start location.

Searches for text after the search start location.

Searches for text that has the same pattern of upper and lower case as the text you stipulate.

Uses a powerful search mechanism, where you specify a template to search for. Using pattern matching, you could search for any character followed by any number, for example.

For more information, click [»](#)

Starts searching again at the other end of the file when the end of the text is reached.
Use this if you want to search the entire file and your cursor is in the middle of the file.

Searches only in the highlighted area.

Locates but does not replace the next occurrence of the text that you specified. You can replace the text when it is found.

Replaces this occurrence of the search text with the text you specified.

Replaces every occurrence of the search text with the text you specified.

Only click this if you are sure that the find/replace text is correct and you want to replace every occurrence.

Repositions the editor at the line number you specify.

Repositions the editor at the line number you specify and closes this dialog box.

Locates files containing the text you stipulate.

Locates files in your project containing the text you stipulate.

Allows you to specify the text to be searched for, and the rules for searching.

Shows a list of the places where the text you specified was found.

Allows you to specify where the system will search for text.

Specifies the text you want to find.

Uses a powerful search mechanism, where you specify a template to search for. Using pattern matching, you could search for any character followed by any number, for example.

For more information, click [»](#)

Searches for text that has the same pattern of upper and lower case as the text you stipulate.

Specifies the folder to search for files containing text. Click **Browse** to select the folder.

Searches the folder you specified and all sub-folders. Sub-folders are folders within folders.

Searches component headers.

Searches in all event handlers and user-defined functions.

Searches the C-library header files included in your project.

Searches for occurrences that are words, and not a part of a larger word.

Searches for all occurrences. Uncheck this to find the first occurrence only.

Lists the files where the text was found. To edit one of the files, select it and click **Edit**.

Lists the places where the text was found. To edit the text, select the occurrence and click **Edit**.

Halts the search.

Opens the selected item for editing in a new edit window.

Starts the search.

Opens the file for editing and closes this dialog box.

Allows you to specify the folder to be searched.

Allows you to choose the target whose source files will be searched.

Specifies the number of spaces that are displayed in place of a `<- tab ->` character.

Specifies the number of spaces that the editor window inserts to automatically indent a block of code.

Replaces tab characters with a number of spaces. Above, type the number of spaces.

Maintains tab characters while editing instead of replacing them with a number of spaces.

Determines whether tab characters are retained in a file or replaced by the number of spaces you specified above.

Resets all tab settings to the defaults.

Colors text based on what it represents (e.g. comment, string).

Shows the different kinds (syntax elements) of text and the colors used to display them. Click on the text representing the syntax element you want to change. Then select the foreground and background colors for that element.

Specifies the name of the color scheme.

Specifies what the text represents (e.g. comment, string).

Changes the colors in the editor window without closing this dialog.

Stores this color scheme. Once stored, you can recall it at a later time. By selecting it from the **Scheme** list.

Erases this color scheme from the list of stored schemes.

Specifies the background color for the current **Syntax element**. Select a color or **Default** from the list.

To specify the colors for a different syntax element, click the sample code window near a character representing the syntax element you want to change or select an element from the list of elements.

Specifies the foreground color for the current **Syntax element**. Select a color or **Default** from the list.

To specify the colors for a different syntax element, click the sample code window near a character representing the syntax element you want to change or select an element from the list of elements.

Specifies the default background color for the editor window. This allows you to change the background color of all syntax elements that use the default background color. Select **Default** from this list to use the standard background color for the system.

Specifies the name that will identify the current color scheme. This allows you to store custom color schemes and change back and forth to different color schemes.

Allows you to insert an object's method. You enter the arguments for the method you choose. If a method is overloaded (multiple parameter lists), you can select the one that you want to use.

This button is only available when you're editing a program (event handler).

Displays the Optima++ Component Library Reference for the selected method or component.

Lists the methods available for each component.

The reference card allows you to paste a method directly into your program. The Parameter Wizard then prompts you for the method's arguments.

Locate the method in the tree and select it. Then click **Parameters** to enter the Parameter Wizard. If the method is overloaded (multiple parameter lists), you can choose the one you want from the list in the Parameter Wizard.

Specifies the class that contains the method you are looking for.

Specifies the method you are searching for.

Enter the full or partial name of the method and press the **Find** button.

Allows you to search for a method by name from a list of all available methods for a class.

Lists the methods that match your search criteria. Use this to put a method into your program or to access the online Reference Manual. Click **Options** to change the way Optima++ searches for a method.

Select the method that you want to use. If the method is overloaded, select the correct version from the box below. Then click **Parameters**.

Allows you to change the way the list is searched.

For example, you can search for all methods containing the letters ooky, or only those methods that start with the letters ooky.

Causes Optima++ to start searching for the method.

Empties the search box and begins a new search.

Describes the way Optima++ will search the list.

Click **Options** to change search method.

Counts the methods that match your search criteria.

Click **Options** to change search method.

Specifies how Optima++ searches for a method.

Choose a method from the drop-down list.

Waits for you to type the search phrase and click **Find Now**.

This method doesn't search while you type.

Updates the search results after each key that you press.
Use this method for incremental or partial searches.

Waits for a pause in your typing before searching.

Allows you to specify what libraries to search.

Enter the variables used for the method's parameters and return value.

Specifies the object whose method you want to call.

Type variables used for the method's parameters here.

Required parameters are arguments of the method that you must specify.

Type the values for optional parameters here.

If you don't specify values for optional parameters, Optima++ uses their default values.

Check this if you want to store the return value in a variable.

Type the name of the variable where you want to store the return value.

No help topic for this button.

This method has more than one version. Select the version of the method that you want to use in your program.

Methods with more than one version are said to be overloaded. You can tell the difference between overloaded methods by their parameter lists.

Lists the versions of the method. Select the version that you want and click **Parameters**.

Methods with more than one version are said to be overloaded. You can tell the difference between overloaded methods by their parameter lists.

Regular expressions

If **Pattern matching** is checked in any Optima++ Find dialog box, Optima++ will interpret the text you type in the **Find** and **Replace** fields as *regular expressions*. A regular expression specifies a pattern of characters for a search or replace operation.

Search strings

The following table lists the strings with special meaning, or *metacharacters*, that you can use in Optima++ search regular expressions.

String	Matches
^	The beginning of a line of text.
\$	The end of a line of text.
.	A period matches any single character. Therefore, ".at" matches bat, cat, eat, and so on.
*	Matches zero or more occurrences of the preceding character. For example, Whe* matches Wh, Whe, Whee, Wheee, and so on.
+	Matches one or more occurrences of the preceding character. For example, Whe+ matches Whe, Whee, Wheee, and so on, but not Wh.
?	Matches zero or one occurrence of the preceding character. For example, Whe? matches either Wh or Whe.
@	Everything after the @ is searched with case sensitivity. For example, @This only matches This, not this or THIS.
~	Everything after the ~ is searched with case insensitivity. For example, ~This matches this, THIS, This, and so on. @ and ~ can be combined within a regular expression to turn on case sensitivity in some areas and turn it off in others.
\	Turns off the special meaning of the next character. For example, \\$ represents a dollar sign character, not the end of the line.

<i>(reg exp)</i>	Use parentheses to treat the enclosed regular expression as a single unit. For example, <i>(abc)*</i> stands for zero or more occurrences of the complete string <i>abc</i> .
!	If ! appears as the first character in a regular expression, all subsequent characters have any special meanings they may have. If ! appears anywhere else in a regular expression, it has no special meaning.
<i>a b</i>	Matches either <i>a</i> or <i>b</i> . For example, <i>(;)\$</i> matches either a semicolon or a blank at the end of a line.
<i>[abc]</i>	Matches any one of the characters inside the square brackets. For example, <i>[be]at</i> matches <i>bat</i> or <i>eat</i> , but not <i>cat</i> .
<i>[a-z]</i>	Matches any one of the characters in the range that appears in the square brackets. For example, <i>[0-9]</i> matches any single digit.
<i>[^abc]</i>	Matches any character <i>except</i> the ones listed in the square brackets. For example, <i>[^be]at</i> matches any string ending in <i>at</i> except <i>bat</i> and <i>eat</i> .
<i>[^a-z]</i>	Matches any character <i>except</i> the ones in the range that appears in the square brackets. For example, <i>[^0-9]</i> matches any character except a digit.

Special replacement strings

In a replace operations with pattern matching, the following replacement strings have special meaning.

String	Replaces
&	Stands for the string matched by the find string. For example, if you ask to replace <i>The</i> with <i>&m</i> , it changes <i>The</i> into <i>Them</i> .
<i>\n</i>	Stands for a linefeed (newline) character.

<code>\t</code>	Stands for a horizontal tab character.
<code>\u</code>	Changes the next item in the replacement string to upper case. For example, <code>\u&</code> stands for the original find string converted into upper case.
<code>\l</code>	Changes the next item in the replacement string to lower case. For example, <code>\l&</code> stands for the original find string converted into lower case.
<code>\U</code>	Changes all following items into upper case until the appearance of <code>\e</code> or <code>\E</code> . For example, <code>\U&&\E</code> stands for two copies of the original find string, both converted into upper case.
<code>\L</code>	Changes all following items into lower case until the appearance of <code>\e</code> or <code>\E</code> . For example, <code>\L&&\E</code> stands for two copies of the original find string, both converted into lower case.
<code>\e</code> or <code>\E</code>	Marks the end of a <code>\U</code> or <code>\L</code> construct.
<code>\</code> <i>number</i>	Refers to the string that matches the <i>number</i> 'th parenthesized part of the find string. For example, suppose the find string is <code>(a)b(c)</code> and the replacement string is <code>\2\1</code> . The result is <code>ca</code> , since <code>\2</code> corresponds to the second parenthesized part <code>(c)</code> and <code>\1</code> corresponds to the first <code>(a)</code> .
<code>\\</code>	Stands for a backslash character.

Notes

Regular expression metacharacters can be combined in many ways. For example:

```
^[0-9]+
```

This stands for any line beginning with one or more digit characters.

One of the most commonly used search regular expressions is `.*` (a period followed by an asterisk) which stands for zero or more characters of any kind. For example:

```
W.*EventData
```

This matches any string of characters beginning with `W` and ending with `EventData`. This would match such strings as

```
WEventData
WDragEventData
WTreeViewEventData
When you see a piece of EventData
```

Regular expressions never match more than one line of text. For example, `W.*EventData` would not match a text string where the `W` appeared on one line and `EventData` appeared on some later line.

Regular expressions always match the longest appropriate string. For example:

```
N.*o
```

This regular expression matches the entire line:

```
Now I have to go to the zoo
```

This occurs instead of matching the shorter pieces:

```
No
Now I have to
Now I have to go
```

Optima++ keyboard shortcuts

Optima++ provides keyboard shortcuts for many operations.

- [Help shortcuts](#)
- [Project shortcuts](#)
- [View shortcuts](#)
- [Searching shortcuts](#)
- [Debugging shortcuts](#)
- [Form design window shortcuts](#)
- [Object inspector shortcuts](#)
- [Editor shortcuts](#)
- [Miscellaneous shortcuts](#)

■ Optima++ keyboard shortcuts

Help shortcuts

F1

Get help about current context

■ Optima++ keyboard shortcuts

Project shortcuts

CTRL+N	Start new project
CTRL+O	Open existing project
CTRL+S	Save current project
F5	Run project
CTRL+F5	Run to cursor

■ Optima++ keyboard shortcuts

View shortcuts


CTRL+>	Go to next window
CTRL+<	Go to previous window
F7	Switch to the code editor window associated with the current form. If the code editor window is already active, switch to the form design window for the active class.
F4	Open Object Inspector
SHIFT+F2	Open Reference Card
SHIFT+F3	Open Classes window
SHIFT+F4	Open Objects window
SHIFT+F5	Open Files window
SHIFT+F6	Open Resources window
SHIFT+F7	Open Targets window
SHIFT+F11	Open Debug Log
SHIFT+F12	Open Error Log
CTRL+SHIFT+F6	Go to previous window

■ Optima++ keyboard shortcuts

Searching shortcuts


CTRL+SHIFT+F3 Find in project

CTRL+SHIFT+F Find in Files

 Optima++ keyboard shortcuts

Debugging shortcuts


SHIFT+F9	Open Breakpoints view
CTRL+L	Open Locals view
CTRL+W	Open Watches view
F10	Step over
F8	Step into
F7	Run to cursor

 Optima++ keyboard shortcuts

Form design window shortcuts

The following shortcuts apply when the form design window is active.


CTRL+C	Copy the selected object(s) to the clipboard
CTRL+X, SHIFT+DEL	Cut the selected object(s) to the clipboard
CTRL+V, SHIFT+INS	Paste any objects held by the clipboard
DEL	Delete the selected object(s)
SHIFT+LEFT/RIGHT	Decrease/Increase the width of the selected component
SHIFT+UP/DOWN	Decrease/Increase the height of the selected component
CTRL+arrow key	Move selected object(s) in the specified direction
CTRL+SHIFT+ARROW	Move faster in the indicated direction
SHIFT+F10	Open context menu for selected object
TAB	Select next object in tabbing order
SHIFT+TAB	Select previous object in tabbing order
Arrow keys	Select next closest object in the specified direction
ENTER	Open Object Inspector
ALT+ENTER	Open selected object's property sheet

 [Optima++ keyboard shortcuts](#)

Object inspector shortcuts

The following shortcuts apply when the Object Inspector is active.

TAB	Cycle from right column to left to combo box at top giving name of object
SHIFT+TAB	Like TAB, but reverse order
ENTER	Accept current settings and move back to form design window
ALT+DOWNARROW	Drop down a combo box showing possible values for right column
CTRL+ENTER	Click the ... button in the right column
Arrow keys	Move one step through list
PAGEUP, PAGEDOWN	Scroll through list
Double-click	Move to next item in combo box or ... button; or if on seperator, resize cloumns to fit values
Other characters	In left column, search for entry beginning with those characters; in right column, Properties page, change value of property

 Optima++ keyboard shortcuts

Editor shortcuts

The following shortcuts apply when the active window is a code editor window.

General shortcuts

F7	Switch to the form design window associated with the current editor window. If the form design window is already active, switch to the code editor window for the active form.
----	--

Run-related shortcuts

CTRL+K	Build (compile) this file
F9	Toggle breakpoint on current line


Text manipulation

CTRL+A	Select all text in the editor window
CTRL+U	Create a new user function
CTRL+P	Print contents of current window
CTRL+Z	Undo previous editing action
CTRL+Y	Redo previously undone action
DEL	Delete selected text
CTRL+C	Copy selected text to the clipboard
CTRL+X	Cut selected text to the clipboard
CTRL+V	Paste text from the clipboard
CTRL+T	Indent current line
CTRL+D	Outdent current line (remove one level of indentation)
F6	Open prototype
SHIFT+F10	Open context menu

Moving through text

CTRL+F	Find/replace in code
F3	Find next occurrence of find string
F2	Open the source corresponding to the selected function name
CTRL+G	Go to line
CTRL+J	Jump to previous cursor location. (Undo Find, F6, F2)
CTRL+B	Go to matching brace
CTRL+]	Go to matching brace
F12	Go to next error found in code
F11	Go to previous error found in code
UPARROW	Go to previous line
DOWNARROW	Go to next line
CTRL+UPARROW	Go to preceding brace in this block
CTRL+DOWNARROW	Go to next brace in this block
PAGEUP	Go up one page
PAGEDOWN	Go down one page
HOME	Go to beginning of line

END	Go to end of line
CTRL+HOME	Go to beginning of code
CTRL+END	Go to end of code
CTRL+TAB	Open the drop-down list of bookmarks
ALT+UPARROW	Go to preceding bookmark
ALT+DOWNARROW	Go to next bookmark
CTRL+M	Add a bookmark on current line
CTRL+SHIFT+M	Go back to most recently created user bookmark

 Optima++ keyboard shortcuts

Miscellaneous shortcuts

F2	Rename (in places where such an operation makes sense; for example, Rename on Classes window renames the selected object)
CTRL+O	Open file in Files window
TAB, SHIFT+TAB	Switch the focus from one part of the window to the other in the Files window and Classes window

