Oracle® Crystal Ball, Fusion Edition

Developer's Guide

RELEASE 11.1.2

ORACLE

ENTERPRISE PERFORMANCE
MANAGEMENT SYSTEM

Crystal Ball Developer's Guide, 11.1.2

Authors: EPM Information Development Team

Contents

1 Welcome

In This Chapter

About the Oracle Crystal Ball Developer Kit

Welcome to the Developer Kit for Oracle Crystal Ball, Fusion Edition.

Using the Crystal Ball Developer Kit, you can automate and control Crystal Ball simulations from within a Visual Basic for Applications (VBA) program. This opens up a whole range of possibilities:

- Running multiple simulations to test different sets of assumptions automatically
- Integrating Crystal Ball with other software tools
- Creating turnkey applications that shield users from program intricacies
- Building custom reports or automate post-simulation analysis
- Setting up specialized simulation environments

The Developer Kit provides a link between Crystal Ball and your application. It consists of a library of macro calls (subroutines and functions) that control many aspects of Crystal Ball. Each copy of Crystal Ball comes enabled to use the Developer Kit, so that programs you develop today can be run by other users as well.

This kit is the key that unlocks the programmability of Crystal Ball and guides you through the many calls available. In addition to the descriptions of the calls, or subroutines and functions, this manual contains examples that illustrate usage of the Developer Kit in several practical applications. You are encouraged to study these examples before starting out on your own applications. Should you need additional help, technical support is available with appropriate licenses.

If you have Oracle Crystal Ball Decision Optimizer, Fusion Edition, Appendix A, "Using the OptQuest Developer Kit," describes how to automate OptQuest optimizations.

See Appendix B, "Using the Predictor Developer Kit," for information about automating and controlling Predictor forecasting.

Note: The Predictor Developer Kit described in this Developer's Guide is completely rewritten. Code written for CB Predictor in Crystal Ball versions earlier than 11.1.1.3.00 is not compatible with the current version of Predictor or this Predictor Developer Kit.

Who Should Use the Crystal Ball Developer Kit

The Crystal Ball Developer Kit is appropriate for advanced users who want to automate repetitive spreadsheet analysis. This manual assumes that readers are familiar with Visual Basic for Applications and Crystal Ball.

What You Will Need

Crystal Ball runs on several versions of Microsoft Windows and Microsoft Excel. For a complete list of required hardware and software, see the *Oracle Crystal Ball Installation and Licensing Guide*.

How This Manual is Organized

This manual contains descriptions of the Crystal Ball Developer Kit macro calls. It also discusses the COM Developer Kit for OptQuest, Predictor, and Oracle Hyperion Smart View for Office, Fusion Edition integration.

Each macro description includes a list of parameters and an example of the use of the relevant calls.

The manual includes the following additional chapters:

- Chapter 2, "Crystal Ball Developer Kit Overview"

  Describes how to use the Crystal Ball Developer Kit and contains an alphabetical list of all Crystal Ball Developer Kit calls with summaries, plus tables of related calls.

- Chapter 3, "Crystal Ball Macro Calls"

  Includes descriptions of all Crystal Ball Developer Kit calls in alphabetical order, with examples.

- Chapter 4, "Changes from Previous Versions"

  Information about changes since previous releases of Crystal Ball.

- Chapter 5, "Crystal Ball Runtime"

  Information about creating applications for people who are not users of Crystal Ball.

- Appendix A, "Using the OptQuest Developer Kit"

Describes how to use the Crystal Ball Decision Optimizer OptQuest Developer Kit to automate OptQuest optimizations.

- Appendix B, "Using the Predictor Developer Kit"

  Describes how to use the Predictor Developer Kit, included with all versions of Crystal Ball, to automate Predictor time-series forecasts.

Documentation Changes

For greater consistency in terminology between the Crystal Ball Developer Kit and VBA:

- The term "macro" describes a sequence of VBA code created using the Crystal Ball Developer Kit.
- The term "function" describes a Developer Kit element that returns a value. For example, the CB.GetCertainty function returns the probability of reaching the specified certainty value.
- The term "subroutine" describes a Developer Kit element that does not return a value — for example, CB.AboutBox.
- The term "call" or "macro call" is used generically to describe any Developer Kit function or subroutine — for example, "The next section describes new Crystal Ball Developer Kit calls."

In this version of the Crystal Ball Developer Kit, all macro calls are listed together in alphabetical order in Chapter 3. For lists of macro calls grouped by functionality, see Chapter 2.

For information on how the Developer Kit calls have changed since the 2000.5 (5.5) version, see Chapter 4, "Changes from Previous Versions".

Technical Support and More

Oracle offers a variety of resources to help you use Crystal Ball, such as technical support, training, and other services. For information, see:

`http://www.oracle.com/crystalball`

2 Crystal Ball Developer Kit Overview

In This Chapter

Introduction

This chapter tells how to use the Crystal Ball Developer Kit and provides lists of macro calls (subroutines and functions) from different categories with references to Chapter 3 for detailed definitions and examples.

The chapter begins with instructions for using the Crystal Ball Developer Kit, followed by an alphabetical list of the Crystal Ball calls, including a brief summary of the actions they perform. This section serves as an index to the Crystal Ball calls in Chapter 3.

The following sections list groups of macro calls used for various purposes:

- "Functions For Use in Microsoft Excel Models" on page 32
- "Opening and Closing Crystal Ball" on page 32
- "Setting Up and Running Simulations" on page 33
- "Controlling Crystal Ball Chart Windows" on page 34
- "Handling Crystal Ball Results" on page 35
- "Setting and Getting Preferences" on page 36
- "Special Calls" on page 37

How to Use the Crystal Ball Developer Kit

This section discusses:

- Calling Crystal Ball From Visual Basic For Applications (VBA) Programs
- Calling Crystal Ball From an External Visual Basic (VB) Program

Note that macros created with the Crystal Ball Developer Kit cannot be placed in spreadsheet cells, although some calls have equivalents (with names ending in FN) that can be used directly in models. For a list of these, see "Functions For Use in Microsoft Excel Models" on page 32.

Calling Crystal Ball From Visual Basic For Applications (VBA) Programs

To call Crystal Ball subroutines and functions from VBA programs, ensure that the following conditions are true:

- Crystal Ball must be running.
- The VBA program must have a reference to Crystal Ball

  .

➤ To create a reference to Crystal Ball from your programs:

1 **Open your VBA program.**

  In Microsoft Excel, for example, select Tools, then Macro, and then Visual Basic Editor.

2 **In the menu, select Tools, then References.**

3 **Look in the Available References list for CBDevKit.**

  - If the reference appears in the list, check it and click OK.
  - If the reference does not appear in the list, continue with step 4.

4 **To locate and load the code module, click Browse.**

5 **Browse to the main Crystal Ball installation folder, by default:**

  `C:\Program Files\Oracle\Crystal Ball`

6 **Change Files Of Type to Excel Files (*.xls;*.xla).**

7 **Double-click CBDevKit.xla.**

  CBDevKit appears in the Available References list with a check in front of it.

  Note: If you upgraded from 2000.x (5.x) versions of the Crystal Ball Developer Kit, locate `cb.xla` in the Available References list and uncheck it.

8 **Click OK.**

  CBDevKit appears as a project where you can access the CB module.

Note: If you do not create a reference to CBDevKit.xla, Crystal Ball calls will not be accessible from your VBA program.

Calling Crystal Ball From an External Visual Basic (VB) Program

Using Crystal Ball from Visual Basic 4.0 or higher is somewhat more complicated than using it from Visual Basic for Applications.

To access Crystal Ball calls, you must first get a reference to the Microsoft Excel application object. To do this, run the following procedure before running Crystal Ball subroutines and functions:

```
Private Sub LoadCB()
   On Error Resume Next
   Set XL = CreateObject("Excel.Application").Parent
   XL.Visible = True
   Set CB = XL.Workbooks("CBDevKit.xla").Modules("CB")
   If Not IsObject(CB) Then
      Set CB = XL.Workbooks.Open(filename:= _
         "C:\Program Files\Oracle\Crystal Ball\CBDevKit.xla")
End If
End Sub
```

If you have stored the file CBDevKit.xla in some other directory than C:\Program Files\Oracle \..., then substitute the appropriate path name.

This procedure gets a reference to the Microsoft Excel application object and puts it into variable XL. Then this variable is used to determine if Crystal Ball is already loaded.

If the Crystal Ball Developer Kit is not loaded, the procedure loads it.

You cannot use Crystal Ball as an OLE automation object. You should call its subroutines and functions through a call to Microsoft Excel's Run macro. The first parameter is the name of a Crystal Ball subroutine or function. Other parameters are passed to the specified Crystal Ball call.

For example the following call to CB.Simulation from a VBA module

It is easy to translate code written in Visual Basic for Applications into Visual Basic code.

```
CB.Simulation 1000, True
```

can be replaced by the VB equivalent

```
XL.Run "CB.Simulation", 1000, True
```

Note: Crystal Ball subroutine and function names are case sensitive when called from outside Microsoft Excel. Be sure to capitalize the CB module name that prefixes the call names. In all other respects, calling Crystal Ball from VB 4.0 is no different from calling it from VBA.

Creating a Microsoft Excel Add-in

➤ To create a Microsoft Excel add-in file:

1. In the VBA editor, select **Tools**, then **Properties**.
2. Select the **Protection** tab.
3. Select **Lock Project For Viewing**.
4. Enter the appropriate password.
5. Click **OK**.
6. In Microsoft Excel, select **File**, then **Save As**.
7. In **Save As Type**, select **Excel Add-In (*.xla)**, the last item.
8. Enter the name to use for the add-in.
9. Click **Save**.

Using Developer Kit Macro Calls in User-defined Macros

The *Oracle Crystal Ball User's Guide* describes how to set up and call user-defined macros from Crystal Ball workbooks. Note that none of the Developer Kit macro calls can be used within user-defined macros such as CBBeforeSimulation. For more information on user-defined macros, see relevant sections at the end of Chapter 5 in the *Oracle Crystal Ball User's Guide*.

A Note About Defaults and Write-Protection

Unless otherwise specified, defaults are pulled from the user's preference files. These defaults can be set through Crystal Ball's preference dialogs or preferences macro calls (such as CB.RunPrefsND) in the Crystal Ball Developer Kit.

Caution! Commands that change Crystal Ball data cells will not run if the cell or the entire worksheet is write-protected. For example, two such commands are CB.Simulation and CB.CellPrefsND. You can use CB.WorksheetProtection to unprotect and reprotect worksheets so certain commands, such as CB.Simulation, will run. For more information on CB.WorksheetProtection, see "CB.WorksheetProtection" on page 230.

Putting Custom Applications into Runtime Mode

If you ever want to deliver an application to someone who has Crystal Ball but is not a skilled user, use the `CB.RuntimeMode` call to hide the Crystal Ball user interface. Then, you can substitute your own controls created with Visual Basic or VBA to activate Crystal Ball features using Crystal Ball Developer Kit calls.

Your application can use any of the Crystal Ball features accessible through the Developer Kit. You can also include Predictor time-series analyses in your applications and users can run simulations in Extreme speed, if it is available to them with Crystal Ball Decision Optimizer.

For details, see Chapter 5, "Crystal Ball Runtime".

Alphabetic List of Crystal Ball Macro Calls

Note: Full descriptions are listed alphabetically in Chapter 3 in the indicated sections.

Table 1 Alphabetic List of Crystal Ball Macro Calls

Name	Description
"CB.AboutBox" on page 42	Displays information about Crystal Ball
"CB.AlertOnArgumentError" on page 43	Displays a message when an argument error occurs
"CB.AlertOnMacroResultError" on page 43	Displays a message when an execution error occurs
"CB.AlertOnObsolete" on page 44	Displays a message when an obsolete call or constant is used
"CB.AssumPrefsND" on page 45	Sets attributes of Crystal Ball assumptions
"CB.AutoDownshift" on page 45	In Crystal Ball Decision Optimizer, determines whether to automatically downshift to Normal speed or fail when a model is incompatible with Extreme speed
"CB.BatchFit" on page 46	Launches the Batch Fit tool
"CB.BatchFitND" on page 47	Runs the Batch Fit tool without displaying dialogs
"CB.Bootstrap" on page 55	Launches the Bootstrap tool
"CB.BootstrapND" on page 55	Runs the Bootstrap tool without displaying dialogs
"CB.CBLoaded" on page 59	Indicates whether Crystal Ball is currently loaded within Microsoft Excel
"CB.CellPrefs" on page 59	Sets attributes of Crystal Ball cells using a dialog
"CB.CellPrefsND" on page 60	Sets attributes of Crystal Ball cells without using a dialog
"CB.ChartPrefs" on page 62	Sets chart preferences for the selected forecast using a dialog
"CB.ChartPrefsND" on page 63	Sets chart preferences for selected forecast without a dialog
"CB.CheckData" on page 65	Checks Crystal Ball data cells for validity, initializes variables with dialogs in case of error
"CB.CheckDataND" on page 66	Checks Crystal Ball data cells for validity, initializes variables without error prompting
"CB.ClearData" on page 68	Erases assumptions, decision variables, and forecasts from selected cells after prompting

Name	Description
"CB.ClearDataND" on page 69	Erases assumptions, decision variables, and forecasts from selected cells without prompting
"CB.CloseAllCharts" on page 70	Closes all open chart windows; equivalent to Analyze, then Close All
"CB.CloseChart" on page 70	Closes the selected chart of the given type
"CB.CloseFore" on page 71	Closes the forecast chart
"CB.CloseSensitiv" on page 72	Closes the selected sensitivity chart
"CB.CloseTrend" on page 73	Closes the selected trend chart
"CB.CopyData" on page 73	Copies Crystal Ball data in the specified range
"CB.CopyDataND" on page 74	Copies selected assumptions, decision variables, or forecasts
"CB.CopyScatter" on page 75	Copies the selected scatter chart to the clipboard
"CB.CopySensitiv" on page 76	Copies the selected sensitivity chart to the clipboard
"CB.CopyTrend" on page 77	Copies the selected trend chart to the clipboard
"CB.CorrelateND" on page 77	Defines a correlation coefficient between two selected assumptions
"CB.CreateChart" on page 79	Creates a new chart of the specified type with the optional name, if specified
"CB.CreateRpt" on page 79	Creates a report from the current simulation and restored results, optimization (with an appropriate license), or time-series forecast using a dialog
"CB.CreateRptND" on page 80	Sets report preferences and creates a report from the current simulation and restored results, optimization (with an appropriate license), or time-series forecast without a dialog
"CB.DataAnalysis" on page 89	Launches the Data Analysis tool
"CB.DataAnalysisND" on page 90	Runs the Data Analysis tool without displaying dialogs
"CB.DecisionTable" on page 94	Launches the Decision Table tool
"CB.DecisionTableND" on page 94	Runs the Decision Table tool without displaying dialogs
"CB.DefineAltParms" on page 97	Creates or changes an alternate parameter set
"CB.DefineAssum" on page 100	Defines or changes assumptions in selected cells with a dialog
"CB.DefineAssumND" on page 101	Defines or changes assumptions in selected cells without a dialog
"CB.DefineAssumFromForeND" on page 105	Defines an assumption from a specified forecast without using a dialog
"CB.DefineDecVar" on page 107	Defines or changes a decision variable in a selected cell using a dialog
"CB.DefineDecVarND" on page 108	Defines or changes a decision variable in a selected cell without a dialog
"CB.DefineFore" on page 111	Defines a forecast in a selected cell using a dialog

Name	Description
“CB.DefineForeND” on page 111	Defines a forecast in a selected cell without a dialog
“CB.DeleteChart” on page 113	Deletes a chart of the specified type and chart ID
“CB.EnumAssum” on page 114	Enumerates assumption cells for all open workbooks
“CB.EnumChart” on page 115	Returns the unique name of the next chart of the specified type
“CB.EnumCorrelation” on page 116	Enumerates correlated assumptions and correlation values for all open workbooks
“CB.EnumDecVar” on page 118	Enumerates decision variable cells for all open workbooks
“CB.EnumFore” on page 119	Enumerates forecast cells for all open workbooks
“CB.ExtractData” on page 120	Extracts data from the current simulation and restored results, optimization (with an appropriate license), or time-series forecast using a dialog
“CB.ExtractDataND” on page 121	Extracts data from the current simulation and restored results, optimization (with an appropriate license), or time-series forecast without a dialog
“CB.Fit” on page 127	Fits a specific probability distribution to a range of data and returns information about the fit
“CB.FormatPrefs” on page 131	Opens the Chart Preferences dialog to set format preferences for the selected forecast
“CB.FormatPrefsND” on page 132	Sets format preferences for the selected forecast without a dialog
“CB.Freeze” on page 134	Defines which Crystal Ball data cells to hold to their spreadsheet values during a simulation using a dialog
“CB.FreezeND” on page 135	Defines which Crystal Ball data cells to hold to their spreadsheet values during a simulation without a dialog
“CB.GetAssum” on page 136	Retrieves information for a specific assumption cell
“CB.GetAssumPercent” on page 138	Calculates the value of the specified assumption that corresponds to the specified percentile
“CB.GetBatchFitOption” on page 140	Returns current settings for the Batch Fit tool
“CB.GetBootstrapOption” on page 143	Returns current settings for the Bootstrap tool
“CB.GetCBAutoLoad” on page 145	Returns the current Crystal Ball autoload setting that indicates whether Crystal Ball is set to open whenever Microsoft Excel opens
“CB.GetCertainty” on page 145	Calculates the certainty level (or probability) of achieving a forecast value at or smaller than a specific threshold
“CB.GetCorrelation” on page 147	Returns the correlation between two assumption cells
“CB.GetDataAnalysisOption” on page 148	Returns current settings for the Data Analysis tool
“CB.GetDecisionTableOption” on page 150	Returns current settings for the Decision Table tool
“CB.GetDecVar” on page 152	Retrieves information for a specific decision variable cell

Name	Description
"CB.GetExcel2007ForegroundMode" on page 153	Returns the current Run Preferences foreground mode setting for Microsoft Excel 2007 in Normal speed
"CB.GetFitParm" on page 154	Returns the value of the specified distribution parameter for the last CB.Fit call
"CB.GetFore" on page 155	Retrieves information for a specific forecast cell
"CB.GetForeData" on page 158	Returns specified forecast data for a given trial
"CB.GetForePercent" on page 158	Returns the value corresponding to a percentile for a specific forecast
"CB.GetForeStat" on page 160	Returns statistics for a specific forecast cell
"CB.GetLockFitParm" on page 162	Returns whether a distribution and, optionally, specific parameters have locked values for fitting purposes
"CB.GetRunPrefs" on page 164	Returns a Run Preferences setting
"CB.GetScenarioAnalysisOption" on page 165	Returns current settings for the Scenario Analysis tool
"CB.GetTwoDSimulationOption" on page 167	Returns current settings for the 2D Simulation tool
"CB.GetVersion" on page 170	Returns the current Crystal Ball version as a full build number or version ID number
"CB.GetWorksheetVersion" on page 171	Returns the Crystal Ball data version of the active or specified worksheet
"CB.IsCBObject" on page 172	Tests if a worksheet or workbook contains Crystal Ball data
"CB.Iterations" on page 172	Returns the number of trials run in a simulation
"CB.LockFitParm" on page 173	Sets parameter locking for distributions and, optionally, specific parameters
"CB.MacroResult" on page 175	Tells you if a Crystal Ball macro call worked correctly
"CB.MacroResultDetail" on page 176	Similar to CB.MacroResult with additional error information
"CB.OpenChart" on page 178	Opens the selected chart of the specified type
"CB.OpenFore" on page 179	Opens a forecast window for the selected cell
"CB.OpenSelection" on page 180	Opens all charts defined for cells in the selected range
"CB.OpenSensitiv" on page 181	Opens the selected sensitivity chart
"CB.OpenTrend" on page 181	Opens the selected trend chart
"CB.PasteData" on page 182	Pastes assumptions, decision variables, or forecasts into selected cells
"CB.Reset" on page 183	Resets the current simulation after prompting
"CB.ResetND" on page 184	Resets the current simulation without confirmation
"CB.RestoreResults" on page 185	Restores simulation results from a file using a dialog
"CB.RestoreResultsND" on page 186	Restores simulation results from a file without a dialog

Name	Description
"CB.RunPrefs" on page 186	Sets Crystal Ball run preferences using a dialog
"CB.RunPrefsND" on page 187	Sets Crystal Ball run preferences without a dialog
"CB.RuntimeMode" on page 191	Hides the Crystal Ball toolbar and menus
"CB.SaveResults" on page 192	Saves current simulation results using a dialog
"CB.SaveResultsND" on page 193	Saves current simulation results without a dialog
"CB.ScatterPrefs" on page 194	Opens the Scatter Preferences dialog for the selected scatter chart
"CB.ScatterPrefsND" on page 194	Sets preferences for the specified scatter chart
"CB.ScenarioAnalysis" on page 198	Opens the Scenario Analysis tool wizard
"CB.ScenarioAnalysisND" on page 198	Runs the Scenario Analysis tool without using dialogs
"CB.SelectAssum" on page 201	Selects all assumptions in the current spreadsheet
"CB.SelectChart" on page 202	Selects the chart with the specified chart ID
"CB.SelectDecVar" on page 203	Selects all decision variables in the current spreadsheet
"CB.SelectFore" on page 203	Selects all forecasts in the current spreadsheet
"CB.SensPrefs" on page 204	Opens the Sensitivity Preferences dialog for the selected sensitivity chart
"CB.SensPrefsND" on page 204	Sets preferences for the specified sensitivity chart
"CB.SetAssum" on page 206	Changes certain attributes for the assumption in a cell
"CB.SetCBAutoLoad" on page 207	Sets or cancels autoloading with Microsoft Excel; determines whether Crystal Ball is set to open whenever Microsoft Excel opens
"CB.SetCBWorkbookPriority" on page 208	Sets the running order for macro applications when multiple workbooks are open
"CB.SetDecVar" on page 209	Changes certain attributes for a decision variable
"CB.SetExcel2007ForegroundMode" on page 211	Sets the Run Preferences foreground mode setting for Microsoft Excel 2007 in Normal speed
"CB.SetFitRange" on page 211	Specifies the data range to be used as input for CB.Fit
"CB.SetFore" on page 212	Changes the settings for a forecast
"CB.SetRange" on page 215	Sets the certainty range for one or all forecasts
"CB.Shutdown" on page 217	Closes Crystal Ball while leaving Microsoft Excel open
"CB.SimResult" on page 217	Returns criteria that stopped the last simulation
"CB.Simulation" on page 218	Runs a simulation for the specified number of iterations
"CB.SingleStep" on page 220	Runs one trial of a simulation, changing assumptions once

Name	Description
"CB.StartMultiSimul" on page 220	Marks the start of multiple simulations
"CB.Startup" on page 222	Loads Crystal Ball into Microsoft Excel if possible and indicates when Crystal Ball has been successfully opened
"CB.StopMultiSimul" on page 222	Marks the end of multiple simulations
"CB.TrendPrefs" on page 223	Opens the Trend Preferences dialog for the selected trend chart
"CB.TrendPrefsND" on page 223	Sets preferences for the specified trend chart
"CB.TwoDSimulation" on page 226	Launches the 2D Simulation tool
"CB.TwoDSimulationND" on page 227	Runs the 2D Simulation tool without showing dialogs
"CB.WorksheetProtection" on page 230	Enables and disables worksheet protection for the active worksheet

Functions For Use in Microsoft Excel Models

The following Crystal Ball functions are also available as spreadsheet functions for use in Microsoft Excel spreadsheet models.

Table 2 Functions For Use in Microsoft Excel Models

Name	Description
CB.GetAssumFN	Retrieves information for a specific assumption cell. See "CB.GetAssum" on page 136.
CB.GetAssumPercentFN	Returns the value corresponding to a percentile for an assumption cell. See "CB.GetAssumPercent" on page 138.
CB.GetCertaintyFN	Returns the certainty level of achieving a forecast value at or smaller than a specific threshold. See "CB.GetCertainty" on page 145.
CB.GetForeDataFN	Returns the value for the given trial for a specific forecast cell. See "CB.GetForeData" on page 158.
CB.GetForePercentFN	Returns the value corresponding to a percentile for a specific forecast. See "CB.GetForePercent" on page 158.
CB.GetForeStatFN	Returns statistic for a specific forecast cell. See "CB.GetForeStat" on page 160.
CB.GetRunPrefsFN	Returns a Run Preference setting. See "CB.GetRunPrefs" on page 164.
CB.IterationsFN	Returns the number of trials run in a simulation. See "CB.Iterations" on page 172.

Opening and Closing Crystal Ball

The macro calls in this group can be used to:

- Load, open, and close Crystal Ball

- Set Crystal Ball autoload defaults
- Determine if Crystal Ball is loaded

The following sections describe macro calls that are used to open and close Crystal Ball:

- "CB.CBLoaded" on page 59
- "CB.GetCBAutoLoad" on page 145
- "CB.SetCBAutoLoad" on page 207
- "CB.Shutdown" on page 217
- "CB.Startup" on page 222

Setting Up and Running Simulations

The macro calls in this group can be used to:

- Identify errors (CB.MacroResult, CB.MacroResultDetail)
- Warn about errors
- Handle Extreme Speed incompatibilities
- Automate the definition of assumptions and forecasts
- Select or copy, paste, or clear Crystal Ball data
- Freeze data cells and run simulations

The following sections describe macro calls that are used to set up and run simulations:

- "CB.AlertOnArgumentError" on page 43
- "CB.AlertOnMacroResultError" on page 43
- "CB.AlertOnObsolete" on page 44
- "CB.AutoDownshift" on page 45
- "CB.ClearData" on page 68
- "CB.ClearDataND" on page 69
- "CB.CopyData" on page 73
- "CB.CopyDataND" on page 74
- "CB.CorrelateND" on page 77
- "CB.DefineAltParms" on page 97
- "CB.DefineAssum" on page 100
- "CB.DefineAssumND" on page 101
- "CB.DefineAssumFromForeND" on page 105
- "CB.DefineDecVar" on page 107
- "CB.DefineDecVarND" on page 108

- “CB.DefineFore” on page 111
- “CB.DefineForeND” on page 111
- “CB.Fit” on page 127
- “CB.Freeze” on page 134
- “CB.FreezeND” on page 135
- “CB.GetFitParm” on page 154
- “CB.GetLockFitParm” on page 162
- “CB.LockFitParm” on page 173
- “CB.MacroResult” on page 175
- “CB.MacroResultDetail” on page 176
- “CB.PasteData” on page 182
- “CB.Reset” on page 183
- “CB.ResetND” on page 184
- “CB.SelectAssum” on page 201
- “CB.SelectDecVar” on page 203
- “CB.SelectFore” on page 203
- “CB.SetAssum” on page 206
- “CB.SetCBWorkbookPriority” on page 208
- “CB.SetDecVar” on page 209
- “CB.SetFitRange” on page 211
- “CB.SetFore” on page 212
- “CB.Simulation” on page 218
- “CB.SingleStep” on page 220
- “CB.StartMultiSimul” on page 220
- “CB.StopMultiSimul” on page 222
- “CB.WorksheetProtection” on page 230

Controlling Crystal Ball Chart Windows

The macro calls in this group can be used to:

- Select which chart windows to open or close
- Open charts for selected cells

The following sections describe calls that are used to control Crystal Ball windows:

- “CB.CloseAllCharts” on page 70
- “CB.CloseChart” on page 70

- “CB.CloseFore” on page 71
- “CB.CloseSensitiv” on page 72
- “CB.CloseTrend” on page 73
- “CB.OpenChart” on page 178
- “CB.OpenFore” on page 179
- “CB.OpenSelection” on page 180
- “CB.OpenSensitiv” on page 181
- “CB.OpenTrend” on page 181

Managing Charts

The macro calls described in the following sections can be used to perform chart management tasks, such as creating charts, copying charts, enumerating or listing charts, and deleting charts:

- “CB.CopyScatter” on page 75
- “CB.CopySensitiv” on page 76
- “CB.CopyTrend” on page 77
- “CB.CreateChart” on page 79
- “CB.DeleteChart” on page 113
- “CB.EnumChart” on page 115
- “CB.SelectChart” on page 202

Handling Crystal Ball Results

The macro calls in this group can be used to:

- Set the certainty range for forecasts
- Get assumption, decision variable, and forecast values, statistics, and percentiles
- Extract data
- Create reports
- Save and restore results

The subroutines and functions described in the following sections are used to work with Crystal Ball results:

- “CB.CreateRpt” on page 79
- “CB.CreateRptND” on page 80
- “CB.ExtractData” on page 120
- “CB.ExtractDataND” on page 121

- "CB.GetAssum" on page 136
- "CB.GetAssumPercent" on page 138
- "CB.GetCertainty" on page 145
- "CB.GetCorrelation" on page 147
- "CB.GetDecVar" on page 152
- "CB.GetFore" on page 155
- "CB.GetForeData" on page 158
- "CB.GetForePercent" on page 158
- "CB.GetForeStat" on page 160
- "CB.RestoreResults" on page 185
- "CB.RestoreResultsND" on page 186
- "CB.SaveResults" on page 192
- "CB.SaveResultsND" on page 193
- "CB.SetRange" on page 215
- "CB.SimResult" on page 217

Setting and Getting Preferences

The macro calls in this group can be used to:

- Set run preferences
- Set format preferences
- Set chart preferences
- Set cell attribute preferences

The subroutines and functions described in the following sections are used to set and get preferences:

- "CB.AssumPrefsND" on page 45
- "CB.CellPrefs" on page 59
- "CB.CellPrefsND" on page 60
- "CB.ChartPrefs" on page 62
- "CB.ChartPrefsND" on page 63
- "CB.FormatPrefs" on page 131
- "CB.FormatPrefsND" on page 132
- "CB.GetExcel2007ForegroundMode" on page 153
- "CB.GetRunPrefs" on page 164
- "CB.RunPrefs" on page 186

- "CB.RunPrefsND" on page 187
- "CB.ScatterPrefs" on page 194
- "CB.ScatterPrefsND" on page 194
- "CB.SensPrefs" on page 204
- "CB.SensPrefsND" on page 204
- "CB.SetExcel2007ForegroundMode" on page 211
- "CB.TrendPrefs" on page 223
- "CB.TrendPrefsND" on page 223

Crystal Ball Tools

The macro calls described in the following sections are used to launch Crystal Ball tools, access tool functionality without displaying dialogs, or return information about tool settings:

- "CB.BatchFit" on page 46
- "CB.BatchFitND" on page 47
- "CB.Bootstrap" on page 55
- "CB.BootstrapND" on page 55
- "CB.DataAnalysis" on page 89
- "CB.DataAnalysisND" on page 90
- "CB.DecisionTable" on page 94
- "CB.DecisionTableND" on page 94
- "CB.GetBatchFitOption" on page 140
- "CB.GetBootstrapOption" on page 143
- "CB.GetDataAnalysisOption" on page 148
- "CB.GetDecisionTableOption" on page 150
- "CB.GetScenarioAnalysisOption" on page 165
- "CB.GetTwoDSimulationOption" on page 167
- "CB.ScenarioAnalysis" on page 198
- "CB.ScenarioAnalysisND" on page 198
- "CB.TwoDSimulation" on page 226
- "CB.TwoDSimulationND" on page 227

Special Calls

The macro calls in this group can be used to:

- Display the About box

- Enumerate assumptions, decision variables, and forecasts
- Determine if a worksheet or workbook contains Crystal Ball data
- Determine how many trials have run in a simulation
- Hide the Crystal Ball user interface in custom applications

The subroutines and functions described in the following sections are included in this group:

3 Crystal Ball Macro Calls

In This Chapter

Introduction

This chapter defines all currently available Crystal Ball Developer Kit macro calls (subroutines and functions). When calls are compared with those in Crystal Ball 2000.5 (5.5), the comparisons usually apply to other 2000.*x* (5.*x*) versions.

For an index with summaries and tables of calls grouped by functionality, see Chapter 2.

Note: Commands that change Crystal Ball data cells will not run if the cell or the entire worksheet is write-protected. For example, two such commands are CB.Simulation and CB.CellPrefsND. You can use CB.WorksheetProtection to unprotect and reprotect worksheets.

CB.AboutBox

This subroutine displays the About Crystal Ball dialog with information about Crystal Ball.

CB.AboutBox Example

This example opens the About Crystal Ball dialog.

```
CB.AboutBox
```

CB.AlertOnArgumentError

Subtopics

- CB.AlertOnArgumentError Example
- CB.AlertOnArgumentError Related Calls

This subroutine displays a message when the value of a subroutine or function argument differs from what is expected.

Table 3 CB.AlertOnArgumentError Parameter

Parameter	VBA Data Type	Description
turnOn	Boolean	True displays an alert message when an argument error occurs; False switches off this alert. The default is True.

CB.AlertOnArgumentError Example

This example switches off alerts when argument errors occur.

```
CB.AlertOnArgumentError False
```

CB.AlertOnArgumentError Related Calls

Name	Description
CB.AlertOnMacroResultError	Displays a message when an execution error occurs
CB.AlertOnObsolete	Displays a message when an obsolete call or constant is used

CB.AlertOnMacroResultError

Subtopics

- CB.AlertOnMacroResultError Example
- CB.AlertOnMacroResultError Related Calls

This subroutine displays a message when an error occurs during the execution of a subroutine or function (macro).

Table 4 CB.AlertOnMacroResultError Parameter

Parameter	VBA Data Type	Description
turnOn	Boolean	True displays an alert message when an error occurs; False switches off this alert. The default is False.

CB.AlertOnMacroResultError Example

This example switches on alerts when macro errors occur.

```
CB.AlertOnMacroResultError True
```

CB.AlertOnMacroResultError Related Calls

Name	Description
CB.AlertOnArgumentError	Displays a message when an argument error occurs
CB.AlertOnObsolete	Displays a message when an obsolete call or constant is used

CB.AlertOnObsolete

Subtopics

This subroutine displays a message when an obsolete call, argument, or constant is used in a subroutine or function (macro).

Table 5 CB.AlertOnObsolete Parameter

Parameter	VBA Data Type	Description
turnOn	Boolean	True displays an alert message when an obsolete call or constant attempts to run; False switches off this alert. The default is True.

CB.AlertOnObsolete Example

This example switches off alerts when obsolete macro calls, arguments, and functions are used.

```
CB.AlertOnObsolete False
```

CB.AlertOnObsolete Related Calls

Name	Description
CB.AlertOnArgumentError	Displays a message when an argument error occurs
CB.AlertOnMacroResultError	Displays a message when an execution error occurs

CB.AssumPrefsND

This subroutine sets attributes of Crystal Ball assumptions without using the Preferences menu on the Define Assumption dialog. The default for any preference is the previous setting in the interface.

Different from Crystal Ball 2000.5 (5.5), this subroutine is used to change preferences for individual assumptions instead of all assumptions. You must select a single cell with an assumption before using this subroutine. If the selected cell does not contain an assumption, a cbErrNoAssumptionInCell error appears.

Table 6 CB.AssumPrefsND Parameters

Parameter	VBA Data Type	Named Constant or Boolean Value	Index Value	Description
PrefItem	Integer	cbAssumShowCDF	1	Sets whether to display the distribution as a PDF (False) or CDF (True)
PrefItem	Integer	cbAssumShowMean	3	Sets whether to display the mean line on the distribution dialog (True)
OnOff	Boolean	True or False	n/a	True turns on the option. False turns off the option.

Note: PrefItem constant cbAssumShowCoordinates [2] is obsolete.

CB.AssumPrefsND Example

This example turns off the marker at the distribution mean for the assumption in cell A1.

```
Range("A1").Select
CB.ND cbAssumShowMean, False
```

CB.AutoDownshift

Subtopics

- CB.AutoDownshift Example 1
- CB.AutoDownshift Example 2

This call specifies whether to automatically "downshift" to Normal speed when a simulation cannot run in Extreme speed and returns the current setting. This setting is global. It is in place for the entire Crystal Ball session. The default setting is True, which switches automatic downshift on. If CB.AutoDownshift is set to False and a simulation cannot run in Extreme speed, the simulation fails and an error is returned. CB.MacroResultDetail ("CB.MacroResultDetail" on page 176) can return the reason for the failure.

Note: If CB.AutoDownshift is not called, Crystal Ball acts as if it has been called with the default setting of True. That is, if a simulation cannot run in Extreme speed, it is automatically downshifted to Normal speed and nothing is recorded in the error log because the simulation ran successfully. You can test whether a simulation can run in Extreme speed by causing the simulation to fail when it cannot run in Extreme speed. To perform the test, explicitly call CB.AutoDownshift and set it to False.

Table 7 CB.AutoDownshift Returned Data Type

Returned Value	Returned Data Type
The current TurnOn parameter setting	Boolean

Table 8 CB.AutoDownshift Parameters

Parameter	VBA Data Type	Named Constant Value	Description
TurnOn	Variant	True or False	True turns on automatic downshifting from Extreme speed to Normal speed. False turns it off.

CB.AutoDownshift Example 1

This example sets Crystal Ball to automatically downshift to Normal speed when a model cannot run in Extreme speed and returns TRUE in cell B2.

```
Range("B2").Value = CB.AutoDownshift(True)
```

CB.AutoDownshift Example 2

To retrieve the current setting of CB.AutoDownshift, use this call in a function in the following form:

```
ReturnValue = CB.AutoDownshift()
```

CB.BatchFit

Subtopics

- CB.BatchFit Example
- CB.BatchFit Related Calls

This call launches the Batch Fit tool wizard. This tool helps define assumptions when you have historical data for a number of variables. The tool automatically selects which probability distribution (binomial, normal, triangular, uniform, and so on) best fits each series of historical data and returns the parameters to use in your model. Batch Fit can also provide a table of goodness-of-fit statistics and a correlation matrix calculated between the historical data.

Note: Before calling CB.BatchFit, reset the simulation.

CB.BatchFit Example

This example resets the Crystal Ball simulation and opens the Batch Fit wizard.

```
CB.ResetND
CB.BatchFit
```

CB.BatchFit Related Calls

Name	Description
CB.BatchFitND	Runs the Batch Fit tool without displaying dialogs
CB.GetBatchFitOption	Returns current settings for the Batch Fit tool

CB.BatchFitND

Subtopics

The Batch Fit tool helps you define assumptions when you have historical data for a number of variables. The tool automatically selects which probability distribution (binomial, normal, triangular, uniform, and so on) best fits each series of historical data and provides you with the parameters to use in your model. Batch Fit can also provide a table of goodness-of-fit statistics and a correlation matrix calculated between the historical data.

This call runs the Batch Fit tool and displays results without using a dialog.

Note: Before calling CB.BatchFitND, reset the Crystal Ball simulation.

Table 9 CB.BatchFitND Parameters

Parameter	Required?	VBA Data Type	Table
Index	Required	Integer	Table 10
Value1	Optional	Variant	Table 11
Value2	Optional	Variant	Table 12
Value3	Optional	Variant	Table 13

Parameter	Required?	VBA Data Type	Table
Value4	Optional	Variant	

Table 10 CB.BatchFitND Index Parameter — Required, Integer

Named Constant Value	Index Value	Description
cbBftInputRange	1	Used with *Value1* to select the location of the data series for fitting, a range address string. The range must be contiguous (adjoining cells)
cbBftInputOrientation	2	Used with *Value1* to specify whether the data cells are in rows or columns
cbBftInputHeader	3	Used with *Value1* to indicate whether the input range has headers. If set to True, there is header text in the top row or first column, depending on the cbBftInputOrientation setting.
cbBftInputLabel	4	Used with *Value1* to indicate whether the input range has labels. If set to True, there is label text in the first column or top row, depending on the cbBftInputOrientation setting.
cbBftSelectDist	5	Used with *Value1* to choose a specific distribution to fit; takes cbDfa[*distribution constant*] as specified in Table 76
cbBftSelectDistAll	6	Chooses all available continuous and discrete distributions to fit
cbBftSelectDistClear	7	Clears all distributions currently selected for fitting
cbBftSelectDistAllContinuous	8	Chooses all available continuous distributions to fit
cbBftSelectDistAllDiscrete	9	Chooses all available discrete distributions to fit
cbBftSelectDistAuto	10	Automatically chooses which distributions to fit
cbBftRankMethod	11	Used with *Value1* to choose a ranking method, as listed in Table 78
cbBftOutputLocation	12	Used with *Value1* to specify an output location in a new workbook
cbBftOutputFormat	13	Used with *Value1* to specify whether output is formatted. If set to True, it is formatted.
cbBftOutputCorrelationMatrix	14	Used with *Value1* to specify whether to output a matrix of correlations between the data series. If set to True, a matrix is produced.
cbBftOutputCorrelation	15	Used with *Value1* to specify whether to correlate assumptions
cbBftLinkToCorrelationMatrix	16	Used with *Value1* to specify whether to link to the matrix of correlations between the data series, assuming cbBftOutputCorrelationMatrix [14] is set to True. If set to True, creates a link to the matrix.
cbBftOutputGoodnessOfFitReport	17	Used with *Value1* to specify whether to output a goodness-of-fit report. If set to True, a report is produced. Value2 can be used to supply a sheetname as a string.
cbBftOutputAssumReport	18	Used with *Value1* to specify whether to output an assumptions report. If set to True, a report is produced. *Value2* can be used to supply a sheetname as a string.
cbBftOutputAssumFullStatistics	19	Used with *Value1* to specify whether to include full statistics in assumptions reports. If set to True, full statistics and percentiles (deciles) are included. Otherwise, only parameter values and assumption charts appear.

Named Constant Value	Index Value	Description
cbBftRun	20	Runs the Batch Fit tool
cbBftUseDistLocking	21	Sets the Batch Fit tool to use parameter locking, the same as checking Lock Parameters on the Fitting Options panel of the Batch Fit wizard
cbBftOutputGoodnessOfFitAllStats	22	Used with *Value1* to indicate that all statistics should appear when a goodness-of-fit report is produced. When set to True, this is the same as checking Show All Goodness-of-Fit statistics on the Output Options panel of the Batch Fit wizard.
cbBftFitDistLockParam	23	Used with *Value1*, *Value2*, *Value3*, and *Value4* to specify which parameter to lock and which value to use
cbBftFitDistLockParamClear	24	Used with *Value1* and *Value2* to specify which parameter to clear
cbBftOutputOrientation	30	Used with *Value1* to specify whether output is in rows or columns
cbBftOutputSheetName	31	Specifies the name of the worksheet containing primary fit results (assumptions) as the string given for *Value1*

Table 11 CB.BatchFitND Value1 Parameter — Optional, Variant

Related Value	Named Constant Value	VBA Data Type	Index Value	Description
For *Index* = 1, cbBftInputRange	A cell reference with format: Range("A1:C300") or "A1:C300"	Variant	n/a	Specifies the range of data to use for fitting
For *Index* = 2, cbBftInputOrientation	cbBftInputRows	Integer	1	Indicates that data cells are in rows
For *Index* = 2, cbBftInputOrientation	cbBftInputColumns	Integer	2	Indicates that data cells are in columns
For *Index* = 3, cbBftInputHeader	True or False	Boolean	n/a	Indicates whether the data range has headers. If set to True, there is header text in the top row or first column, depending on the cbBftInputOrientation setting.
For *Index* = 4, cbBftInputLabel	True or False	Boolean	n/a	Indicates whether the data range has labels. If set to True, there is label text in the first column or top row, depending on the cbBftInputOrientation setting.
For *Index* = 5, cbBftSelectDist	cbDfa... from Table 77	Integer	0-23, 50-53	Indicates a specific distribution or distribution type(s) to fit to the data
For *Index* = 11, cbBftRankMethod	Methods listed in Table 78	Integer	1-5	Specifies a ranking method, as listed in Table 78
For *Index* = 12, cbBftOutputLocation	cbBftNewWorkbook	Integer	1	Used with *Value2* to indicate whether results should go in a new workbook

Related Value	Named Constant Value	VBA Data Type	Index Value	Description
For *Index* = 12, cbBftOutputLocation	cbBftNewWorksheet	Integer	3	Used with *Value2* to indicate whether results should go in a new worksheet
For *Index* = 13, cbBftOutputFormat	True or False	Boolean	n/a	Indicates whether results should be formatted; if True, formatting is applied
For *Index* = 14, cbBftOutputCorrelationMatrix	True or False	Boolean	n/a	Indicates whether to create a correlation matrix; if True, the matrix is produced
For *Index* = 15, cbBftOutputCorrelation	True or False	Boolean	n/a	Indicates whether to correlate assumptions; if True, assumptions are correlated. *Value2* can be used to indicate a correlation threshold.
For *Index* = 16, cbBftLinkToCorrelationMatrix	True or False	Boolean	n/a	Indicates whether to link to the matrix of correlations between the data series; if True, creates the link
For *Index* = 17, cbBftOutputGoodness OfFitReport	True or False	Boolean	n/a	Indicates whether to output a goodness-of-fit report; if True, the report is produced
For *Index* = 18, cbBftOutputAssumReport	True or False	Boolean	n/a	Indicates whether to output an assumption report; if True, the report is produced
For *Index* = 19, cbBftOutputAssumFullStatistics	True or False	Boolean	n/a	Indicates whether to include full statistics with an assumption report; if True, the report contains full statistics, including deciles
For *Index* = 21, cbBftUseDistLocking	True or False	Boolean	n/a	Indicates whether the Batch Fit tool should use parameter locking; if True, parameter locking is set (the same as checking Lock Parameters on the Fitting Options panel of the Batch Fit wizard)
For *Index* = 22, cbBftOutputGoodnessOf FitAllStats	True or False	Boolean	n/a	Indicates whether all statistics should appear when a Goodness of Fit report is produced, the same as checking Show All Goodness-of-Fit statistics on the Output Options panel of the wizard; if set to True, all statistics appear

Related Value	Named Constant Value	VBA Data Type	Index Value	Description
For *Index* = 23, cbBftFitDistLockParam	The name of a lockable distribution from Table 53; see Description	Integer	The index number of a lockable distribution from Table 53; see Description	Specifies one of the following distributions whose parameters are locked. The following distributions from Table 53 can be used as *Value1* parameters: ● Gamma distribution — Location, Shape parameters ● Lognormal distribution — Location parameter ● Student's *t* distribution — Degrees of Freedom parameter ● Weibull distribution — Location, Shape parameters ● Binomial distribution — Trials parameter ● Hypergeometric distribution — Trials parameter Use *Value2* to specify the number of the target parameter.
For *Index* = 24, cbBftFitDistLockParamClear	The name of a lockable distribution from Table 53; see Description	Integer	The index number of a lockable distribution from Table 53; see Description	Specifies one of the following distributions whose parameters are cleared. The following distributions from Table 53 can be used as *Value1* parameters: ● Gamma distribution — Location, Shape parameters ● Lognormal distribution — Location parameter ● Student's *t* distribution — Degrees of Freedom parameter ● Weibull distribution — Location, Shape parameters ● Binomial distribution — Trials parameter ● Hypergeometric distribution — Trials parameter Use *Value2* to specify the number of the target parameter.
For *Index* = 30, cbBftOutputOrientation	cbBftOutputRows	Integer	1	Indicates that output cells are in rows; equivalent of selecting **Fill to the right** in the user interface
For *Index* = 30, cbBftOutputOrientation	cbBftOutputColumns	Integer	2	Indicates that output cells are in columns; equivalent of selecting **Fill downward** in the user interface

Table 12 CB.BatchFitND Value2 Parameter — Optional, Variant

Related Value	Named Constant Value	VBA Data Type	Index Value	Description
For *Index* = 12, cbBftOutputLocation	True or False	Boolean	n/a	Used with *Value1* = cbBftNewWorkbook or cbBftNewWorksheet to indicate whether the output goes in a new workbook or worksheet. If cbBftNewWorksheet is set to False, *Value3* can be used to indicate the first (upper left) cell of the output range on the current worksheet. Otherwise, cell A1 is assumed.
For *Index* = 15, cbBftOutputCorrelation	A whole number or decimal between 0 and 1, inclusive	Double	n/a	Used with *Value1* = True to indicate that assumptions should be correlated if the absolute value of their correlation is equal to or greater than the specified threshold
For *Index* = 17, cbBftOutputGoodnessOfFitReport	A string that indicates the report name	String	n/a	Used with *Value1* = True to specify a report name string
For *Index* = 18, cbBftOutputAssumReport	A string that indicates the report name	String	n/a	Used with *Value1* = True to specify a report name string
For *Index* = 23, cbBftFitDistLockParam	The index of a lockable parameter from Table 53; see Description	Integer	The index number of a lockable parameter for the distribution specified in Value1 (all from Table 53; see Description)	Specifies one of the following distribution parameters to lock. The following from Table 53 can be used as *Value2* parameters: ● Gamma distribution — Location, Shape parameters ● Lognormal distribution — Location parameter ● Student's *t* distribution — Degrees of Freedom parameter ● Weibull distribution — Location, Shape parameters ● Binomial distribution — Trials parameter ● Hypergeometric distribution — Trials parameter Use *Value3* to specify the value of the target parameter.

Related Value	Named Constant Value	VBA Data Type	Index Value	Description
For *Index* = 24, cbBftFitDistLockParamClear	The index of a lockable parameter from Table 53; see Description	Integer	The index number of a lockable parameter for the distribution specified in Value1 (all from Table 53; see Description)	Specifies one of the following distribution parameter(s) to clear. The locking settings and values are cleared. The following from Table 53 can be used as *Value2* parameters: ● Gamma distribution — Location, Shape parameters ● Lognormal distribution — Location parameter ● Student's *t* distribution — Degrees of Freedom parameter ● Weibull distribution — Location, Shape parameters ● Binomial distribution — Trials parameter ● Hypergeometric distribution — Trials parameter

Table 13 CB.BatchFitND Value3 Parameter — Optional, Variant

Related Value	Named Constant Value	VBA Data Type	Value	Description
For *Index* = 23, cbBftFitDistLockParam	n/a	Double	A number within the acceptable value range for the specified parameter according to Table 53	The actual value to be used for the specified parameter. Cell references cannot be used here.

Note: Before you call CB.BatchFitND, reset the Crystal Ball simulation.

CB.BatchFitND Example 1

This example resets the Crystal Ball simulation and fits all distributions to a columnar data set in cells A1 through C500 (including a header row). The selected ranking method is Anderson-Darling. Results are output in a range on the current worksheet, where D1 is the first cell (in the upper left corner). Resulting assumption correlations are displayed in a matrix. Correlations are established for assumptions whose absolute correlation is equal to or greater than 0.2. Goodness-of-fit and assumption reports are created with specified names. The assumption report includes full statistics. The last line of code runs the Batch Fit tool.

```
CB.ResetND

' Select all distributions
CB.BatchFitND cbBftSelectDistAll

' Set input range orientation (data in columns)
CB.BatchFitND cbBftInputOrientation, cbBftInputColumns
```

```
' Input range has headers
CB.BatchFitND cbBftInputHeader, True

' Select input range
CB.BatchFitND cbBftInputRange, "A1:C500"

' Select the Anderson-Darling ranking method
CB.BatchFitND cbBftRankMethod, cbFitAndersonDarling

' Select output location (starting in cell D1 of a new worksheet)
CB.BatchFitND cbBftOutputLocation, cbBftNewWorksheet, False, "D1"

' Show correlation matrix between the data series
CB.BatchFitND cbBftOutputCorrelationMatrix, True

' Do not format output
CB.BatchFitND cbBftOutputFormat, False

' Correlate assumptions where the absolute correlation is 0.2 or greater
CB.BatchFitND cbBftOutputCorrelation, True, 0.2
'Output results to "Batch Fit Results"
CB.BatchFitND cbBftOutputSheetName, "Batch Fit Results"
'Fill results downward
CB.BatchFitND cbBftOutputOrientation, cbBftOutputColumns

' Create goodness-of-fit report named "Fit Statistics"
CB.BatchFitND cbBftOutputGoodnessOfFitReport, True, "Fit Statistics"

' Create assumption report, "Fit Assumption Report", with full statistics
CB.BatchFitND cbBftOutputAssumReport, True, "Fit Assumption Report"

CB.BatchFitND cbBftOutputAssumFullStatistics, True

' Run the tool
CB.BatchFitND cbBftRun
```

CB.BatchFitND Example 2

This example runs the Batch Fit tool using default preferences or settings from a previous run in the current Microsoft Excel session.

```
CB.ResetND

' Run Batch Fit
CB.BatchFitND cbBftRun
```

CB.BatchFitND Related Calls

Name	Description
CB.BatchFit	Launches the Batch Fit tool
CB.GetBatchFitOption	Returns current settings for the Batch Fit tool

CB.Bootstrap

Subtopics

- CB.Bootstrap Example
- CB.Bootstrap Related Calls

This call launches the Bootstrap tool wizard. This tool estimates the reliability (accuracy) of forecast statistics and other sample data. The Bootstrap tool displays a forecast chart of the distributions for each statistic, capability metric, or percentile and creates a workbook summarizing the data.

Note: Before calling CB.Bootstrap, reset the simulation.

CB.Bootstrap Example

This example resets the Crystal Ball simulation and opens the Bootstrap wizard.

```
CB.ResetND
CB.Bootstrap
```

CB.Bootstrap Related Calls

Name	Description
CB.BootstrapND	Runs the Bootstrap tool without displaying dialogs
CB.GetBootstrapOption	Returns current settings for the Bootstrap tool

CB.BootstrapND

Subtopics

- CB.BootstrapND Example 1
- CB.BootstrapND Example 2
- CB.BootstrapND Related Calls

The Bootstrap tool estimates the reliability (accuracy) of forecast statistics, process capability metrics, percentiles, and other sample data. This call runs the Bootstrap tool and displays results without using a dialog.

Note: Before calling CB.BootstrapND, reset the Crystal Ball simulation.

Table 14 CB.BootstrapND Parameters

Parameter	Required?	VBA Data Type	Table
Index	Required	Integer	Table 15
Index2	Optional	Integer	Table 16
Value	Optional	Variant	Table 17

Table 15 CB.BootstrapND Index Parameter – Required, Integer

Named Constant Value	Index Value	Description
cbBtsChooseTarget	1	Sets the target forecast for the bootstrap operation (operates on the forecast in the currently selected cell)
cbBtsChooseDataRange	2	Selects the data values for the bootstrap operation (operates on the currently selected cell range)
cbBtsMethod	3	Used with *Index2* to specify a bootstrap method
cbBtsAnalyze	4	Used with *Index2* to set bootstrap analysis options
cbBtsPercent	5	Used with *Index2* to specify percentile settings
cbBtsSample	6	Used with *Index2* to set sample options
cbBtsForeOption	7	Used with *Index2* to set forecast chart window display options
cbBtsRun	8	Runs the Bootstrap tool

Table 16 CB.BootstrapND Index2 Parameter – Optional, Integer

Related Value	Named Constant Value	Index Value	Description
For *Index* = 3, cbBtsMethod: Integer	cbBtsMethodResample	1	Runs one simulation and resamples the data (faster but less accurate)
For *Index* = 3, cbBtsMethod: Integer	cbBtsMethodMultiSim	2	Runs multiple simulations and creates diverse data (slower but more accurate)
For *Index* = 4, cbBtsAnalyze: Integer	cbBtsAnalyzeStat	1	Analyzes distributions of statistics
For *Index* = 4, cbBtsAnalyze: Integer	cbBtsAnalyzePercent	2	Analyzes distributions of percentiles
For *Index* = 4, cbBtsAnalyze: Integer	cbBtsAnalyzeCapMetrics	3	Analyzes distributions of process capability metrics (only available if process capability metrics are activated on the Statistics tab of the Crystal Ball Run Preferences dialog or using the CB.RunPrefsND call

Related Value	Named Constant Value	Index Value	Description
For *Index* = 5, cbBtsPercent: Integer	cbBtsPercentTenNinety	1	Specifies the 10th and 90th percentiles
For *Index* = 5, cbBtsPercent: Integer	cbBtsPercentFiveNinetyFive	2	Specifies the 5th and 95th percentiles
For *Index* = 5, cbBtsPercent: Integer	cbBtsPercentQuartiles	3	Specifies quartiles (25%-ile increments from 0 to 100, inclusive)
For *Index* = 5, cbBtsPercent: Integer	cbBtsPercentQuintiles	4	Specifies quintiles (20%-ile increments from 0 to 100, inclusive)
For *Index* = 5, cbBtsPercent: Integer	cbBtsPercentDeciles	5	Specifies deciles (10%-ile increments from 0 to 100, inclusive)
For *Index* = 5, cbBtsPercent: Integer	cbBtsPercentIcosatiles	6	Specifies icosatiles (5%-ile increments from 0 to 100, inclusive)
For *Index* = 5, cbBtsPercent: Integer	cbBtsPercentCustom	7	Defines custom percentiles, supplied by an optional *Value* entry (a one-dimensional cell range containing percentiles). If *Value* is omitted, the default percentiles are 2.5, 5, 50, 95, 97.5.
For *Index* = 6, cbBtsSample: Integer	cbBtsNumberSamples	1	Used with a *Value* entry to specify the number of bootstrap samples to use
For *Index* = 6, cbBtsSample: Integer	cbBtsTrialsPerSample	2	Used with a *Value* entry to specify the number of trials to run for each bootstrap sample
For *Index* = 7, cbBtsForeOption: Integer	cbBtsShowDefinedFore	1	Uses display settings for each forecast chart; the equivalent of the Show Forecasts As Defined dialog setting
For *Index* = 7, cbBtsForeOption: Integer	cbBtsShowTargetFore	2	Shows only the target forecast; the equivalent of the Show Only Target Forecast dialog setting
For *Index* = 7, cbBtsForeOption: Integer	cbBtsHideFore	3	Hides all forecasts; the equivalent of the Hide All Forecasts dialog setting

Table 17 CB.BootstrapND Value Parameter – Optional, Variant

Related Value	Named Constant Value	VBA Data Type	Index Value	Description
For *Index2* = cbBtsPercentCustom	A string containing a Microsoft Excel range that holds desired percentile values. The values must be entered into Microsoft Excel as numbers, not formatted as percentages; can be decimals.	String	n/a	Specifies custom percentile values to analyze
For *Index2* = cbBtsNumberSamples	A positive whole number	Integer	n/a	Number of bootstrap samples to use
For *Index2* = cbBtsTrialsPerSample	A positive whole number	Long Integer	n/a	Number of trials to run for each bootstrap sample

CB.BootstrapND Example 1

This example resets the Crystal Ball simulation, selects a target forecast in cell A1, specifies the multisimulation method, specifies analysis of custom percentiles (included in the range F17:H17), specifies 100 bootstrap samples with 1000 trials per sample, indicates that only the target forecast will display, and then runs the Bootstrap tool.

```
CB.ResetND

' Select target forecast
[A1].Select  '<- A1 must contain a forecast
CB.BootstrapND cbBtsChooseTarget

' Select multisimulation method
CB.BootstrapND cbBtsMethod, cbBtsMethodMultiSim

' Select what to analyze -- percentiles, in this case
CB.BootstrapND cbBtsAnalyze, cbBtsAnalyzePercent

' Select custom percentiles in the specified cell range
CB.BootstrapND cbBtsPercent, cbBtsPercentCustom, "F17:H17"

' Set up sampling options; 100 samples with 1000 trials per sample
CB.BootstrapND cbBtsSample, cbBtsNumberSamples, 100
CB.BootstrapND cbBtsSample, cbBtsTrialsPerSample, 1000

' Set display settings to show only the target forecast
CB.BootstrapND cbBtsForeOption, cbBtsShowTargetFore

' Run Bootstrap
CB.BootstrapND cbBtsRun
```

CB.BootstrapND Example 2

This example runs the Bootstrap tool using default preferences or settings from a previous run in the current Microsoft Excel session.

```
CB.ResetND

' Run Bootstrap
CB.BootstrapND cbBtsRun
```

CB.BootstrapND Related Calls

Name	Description
CB.Bootstrap	Launches the Bootstrap tool
CB.GetBootstrapOption	Returns current settings for the Bootstrap tool.

CB.CBLoaded

Subtopics

- CB.CBLoaded Example
- CB.CBLoaded Related Calls

This function determines if Crystal Ball is loaded within Microsoft Excel.

Note: CB.CBLoaded must return True before any Crystal Ball features can be used.

Table 18 CB.CBLoaded Returned Data Type

Returned Value	Returned Data Type
Whether Crystal Ball is loaded within Microsoft Excel	Boolean

CB.CBLoaded Example

To retrieve the current setting of CB.CBLoaded, use this call in a function as follows:

```
ReturnValue = CB.CBLoaded()
```

CB.CBLoaded Related Calls

Name	Description
CB.GetCBAutoLoad	Returns the current Crystal Ball autoload setting that indicates whether Crystal Ball is set to open whenever Microsoft Excel opens
CB.SetCBAutoLoad	Sets or cancels autoloading with Microsoft Excel; determines whether Crystal Ball is set to open whenever Microsoft Excel opens
CB.Shutdown	Closes Crystal Ball while leaving Microsoft Excel open
CB.Startup	Loads Crystal Ball into Microsoft Excel if possible and indicates when Crystal Ball has been successfully opened

CB.CellPrefs

Subtopics

- CB.CellPrefs Example
- CB.CellPrefs Related Calls

This subroutine opens the Cell Preferences dialog so that you can set cell preferences for existing Crystal Ball data cells or new cells.

CB.CellPrefs Example

This example opens the Cell Preferences dialog so you can set and apply new cell preferences.

```
CB.CellPrefs
```

CB.CellPrefs Related Calls

Name	Description
CB.CellPrefsND	Sets attributes of Crystal Ball cells

CB.CellPrefsND

Subtopics

This subroutine sets attributes of Crystal Ball cells without the Cell Preferences dialog. The default for any preference is the previous setting in the Cell Preferences dialog.

Caution! CB.CellPrefsND cannot run if a cell with Crystal Ball data is located in a write-protected worksheet.

Table 19 CB.CellPrefsND Parameters

Parameter	VBA Data Type	Description
Index1	Integer	Changes assumption, forecast, or decision variable cells depending on the selected constant or index value: ● cbCelAssum = 1 ● cbCelFore = 2 ● cbCelDecVar = 3
Index2	Integer	Sets preferences depending on the Value parameter that follows it. For constant and index values, see Table 20
Value	Variant	Works with Index2 to specify preference settings, as described in Table 21
DoNotApply (Optional)	Boolean	True applies changes only to new Crystal Ball cells. False applies changes to existing and new Crystal Ball cells. The default is False.

Parameter	VBA Data Type	Description
Level (Optional)	Integer	Specifies whether changes are applied globally to all open and new workbooks, to cells of that type in the active workbook, or to cells of that type in the active worksheet, depending on the selected constant or index value: ● cbCelGlobal = 0 ● cbCelWorkbook = 1 ● cbCelWorksheet = 2

The *Index2* parameter listed in the following table (Table 20) works with the *Value* parameter to set cell preferences. For *Value* parameter values, see Table 21.

Table 20 CB.CellPrefsND Index2 Parameter Values — Integer, Required

Constant Value	Index Value	Description
cbCelPattern	1	Sets a pattern choice defined by *Value* (Table 21)
cbCelColor	2	Sets a color choice defined by *Value*
cbCelNote	3	Turns notes on and off, depending on *Value*
cbCelDistMean	4	Sets the value in assumption cells to be the mean of the defined distribution
cbCelRangeMidpoint	4	Sets the value in decision variable cells to be the midpoint of the defined range
cbCelDistMedian	5	Sets the value in assumption cells to be the median of the defined distribution
cbCelRangeMin	6	Sets the value in decision variable cells to be the minimum of the defined range
cbCelRangeMax	7	Sets the value in decision variable cells to be the maximum of the defined range

The *Value* parameter values listed in the following table work with the *Index2* parameter values to set cell preferences. For *Index2* values, see Table 20.

Table 21 CB.CellPrefsND Value Parameter Values — Variant, Required

Used With Specified Values of Index2	Named Constant or Boolean Value	Index Value	Description
For *Index2* = 1, cbCelPattern: Integer	n/a	0 through 17	0 specifies no pattern; 1 through 17 specify available patterns
For *Index2* = 2, cbCelColor: Integer	n/a	0 to 15	0 specifies no color; 1-15 specify available colors
For *Index2* = 3, cbCelNote: Boolean	True or False	n/a	True turns on notes; False turns off notes
For *Index2* = 4, cbCelDistMean: Boolean	True or False	n/a	For assumption cells, True changes the cell value to the mean; False leaves the existing values in the cells
For *Index2* = 4, cbCelRangeMidpoint: Boolean	True or False	n/a	For decision variable cells, True changes the cell value to the midpoint; False leaves the existing values in the cells

Used With Specified Values of Index2	Named Constant or Boolean Value	Index Value	Description
For *Index2* = 5, cbCelDistMedian: Boolean	True or False	n/a	For assumption cells, True changes the cell value to the median; False leaves the existing values in the cells
For *Index2* = 6, cbCelRangeMin: Boolean	True or False	n/a	For decision variable cells, True changes the cell value to the range minimum; False leaves the existing values in the cells
For *Index2* = 7, cbCelRangeMax: Boolean	True or False	n/a	For decision variable cells, True changes the cell value to the range maximum; False leaves the existing values in the cells

CB.CellPrefsND Example

This example changes cell preferences for assumption cells, including both existing and future assumptions. It turns off any pattern, changes the color to cyan, and displays assumption cell notes.

```
CB.CellPrefsND cbCelAssum, cbCelPattern, 0, False
CB.CellPrefsND cbCelAssum, cbCelColor, 8, False
CB.CellPrefsND cbCelAssum, cbCelNote, True, False
```

CB.CellPrefsND Related Calls

Name	Description
CB.CellPrefs	Displays the Cell Preferences dialog

CB.ChartPrefs

Subtopics

This subroutine opens the Chart Preferences dialog for the selected assumption or forecast cell so you can change the chart preferences. If an assumption or forecast cell is not selected, error code -5102 appears (from CB.MacroResultDetail.NewValue).

Before calling this subroutine to change the chart preferences for a particular assumption or forecast chart, select the corresponding cell.

Note: Requirements for this subroutine are slightly different than for Crystal Ball 2000.5 (5.5). For 2000.*x* (5.*x*) versions, you could use this subroutine without selecting a cell to set default forecast chart settings.

CB.ChartPrefs Example

This example runs a simulation of 500 trials, selects cell B7, and then opens the Chart Preferences dialog.

```
CB.Simulation 500
Range("B7").Select
CB.ChartPrefs
```

CB.ChartPrefs Related Calls

Name	Description
CB.ChartPrefsND	Sets chart preferences for the forecast or assumption corresponding to the selected cell without displaying the Chart Preferences dialog
CB.Simulation	Runs a Crystal Ball simulation

CB.ChartPrefsND

Subtopics

This subroutine sets chart preferences for the forecast or assumption corresponding to the selected cell without displaying the Chart Preferences dialog. The default for any preference is the previous setting in the Chart Preferences dialog.

Note: You must select a forecast or assumption cell before calling this subroutine to change the chart preferences for a single forecast or assumption. If you call this subroutine to change a single forecast or assumption chart (*All* is False) when no forecast or assumption cell is selected, CB.MacroResultDetail.NewValue returns error code -5102.

Table 22 CB.ChartPrefsND Parameters

Parameter	VBA Data Type	Description
Index	Integer	Sets basic forecast and assumption chart preferences, according to the *Value parameter settings*. For constant and index values, see Table 23.
Value	Variant	Works with the Index parameter to set forecast and assumption chart preferences. For available value settings, see Table 24.
All (Optional)	Variant	True applies settings to all assumptions or forecasts. False applies settings only to the selected assumption or forecast. The default is False.

Note: Optional parameters SeriesAxisIndex, Value2, and Value3 are currently not supported.

The *Index* parameter (Table 23) works with the *Value* parameter (Table 24) to set basic forecast and assumption chart preferences.

Table 23 CB.ChartPrefsND Index Parameter Values — Required, Integer

Named Constant Value	Index Value	Description
cbChtChartType	1	Sets the chart type, according to the *Value* setting
cbChtDistType	2	Sets the distribution type, according to the *Value* setting
cbChtGroups	3	Sets the grouping in the histogram, according to the *Value* setting
cbChtMeanLine	5	Turns on and off the mean line, according to the *Value* setting
cbChtHorizGrids	6	Turns on and off horizontal grid lines, according to the *Value* setting

Note: The following Index constant is obsolete in Crystal Ball 7.*x* and later (including 11.*x*): cbChtRedrawFrequency, Index = 4.

The *Value* parameter (Table 24) works with the *Index* parameter (Table 23) to set forecast and assumption chart preferences.

Table 24 CB.ChartPrefsND Value Parameter Values — Required Variant

Used With Specified Values of Index	Named Constant or Boolean Value	Index Value	Description
For *Index* = 1, cbChtChartType: Integer	cbTypArea	1	Changes the assumption or forecast chart type to an area chart
For *Index* = 1, cbChtChartType: Integer	cbTypOutline	2	Changes the assumption or forecast chart type to an outline chart
For *Index* = 1, cbChtChartType: Integer	cbTypColumn	3	Changes the assumption or forecast chart type to a column chart
For *Index* = 2, cbChtDistType: Integer	cbDstFrequency	1	Changes the distribution type to a frequency distribution
For *Index* = 2, cbChtDistType: Integer	cbDstCumulative	2	Changes the distribution type to a cumulative frequency distribution
For *Index* = 2, cbChtDistType: Integer	cbDstReverseCumul	3	Changes the distribution type to a reverse cumulative frequency distribution
For *Index* = 3, cbChtGroups: Boolean	cbGrp15	1	Equivalent to the Chart Bins Density setting of Lowest on the General tab of the Chart Preferences dialog
For *Index* = 3, cbChtGroups: Boolean	cbGrp25	2	Equivalent to the Chart Bins Density setting of Low on the General tab of the Chart Preferences dialog

Used With Specified Values of Index	Named Constant or Boolean Value	Index Value	Description
For *Index* = 3, cbChtGroups: Boolean	cbGrp50	3	Equivalent to the Chart Bins Density setting of Low on the General tab of the Chart Preferences dialog
For *Index* = 3, cbChtGroups: Boolean	cbGrp75	4	Equivalent to the Chart Bins Density setting of Medium on the General tab of the Chart Preferences dialog
For *Index* = 3, cbChtGroups: Boolean	cbGrp100	5	Equivalent to the Chart Bins Density setting of High on the General tab of the Chart Preferences dialog
For *Index* = 3, cbChtGroups: Boolean	cbGrp150	6	Equivalent to the Chart Bins Density setting of High on the General tab of the Chart Preferences dialog
For *Index* = 3, cbChtGroups: Boolean	cbGrp300	7	Equivalent to the Chart Bins Density setting of Highest on the General tab of the Chart Preferences dialog
For *Index* = 5, cbChtMeanLine: Boolean	True or False	n/a	Turns on (True) or off (False) the mean line
For *Index* = 6, cbChtHorizGrids: Boolean	True or False	n/a	Turns on (True) or off (False) the horizontal grid lines

Note: See the *Oracle Crystal Ball User's Guide* for more information on how to customize assumption and forecast charts.

CB.ChartPrefsND Example

This example selects a forecast cell and then changes its forecast chart to show a mean line.

```
Range("B7").Select
CB.ChartPrefsND cbChtMeanLine, True
```

CB.ChartPrefsND Related Calls

Name	Description
CB.ChartPrefs	Displays the Chart Preferences dialog

CB.CheckData

Subtopics

This subroutine scans Crystal Ball data on open worksheets, checking the validity of all Crystal Ball data objects, including assumptions, decision variables, forecasts, and correlations. Then,

it initializes internal variables. If errors occur, appropriate dialogs appear and you are prompted to correct them.

You should call CB.CheckData or CB.CheckDataND each time you start a new enumeration, use a CB.Get... call that relies on cell references that might have changed, and each time you delete or add data. You should use CB.CheckData or CB.CheckDataND before using calls that extract data or create reports.

CB.CheckData Example

This example runs a simulation of 500 trials, initializes all the internal variables for the following enumeration, and then enumerates all the forecasts, opening each forecast chart.

```
Dim s As String
Dim r As Range
CB.Simulation 500
CB.CheckData
s = CB.EnumFore
While s <> ""
   Set r = Range(s)
   CB.OpenFore
   s = CB.EnumFore
   t = t + 1
Wend
```

CB.CheckData Related Calls

Name	Description
CB.CheckDataND	Checks Crystal Ball data cells for validity, initializes variables without error prompting
CB.EnumAssum	Enumerates assumption cells for all open workbooks
CB.EnumFore	Enumerates forecast cells for all open workbooks
CB.EnumDecVar	Enumerates decision variable cells for all open workbooks

CB.CheckDataND

Subtopics

Like CB.CheckData, this subroutine scans Crystal Ball data on open worksheets, checking the validity of all Crystal Ball data objects, including assumptions, decision variables, forecasts, and correlations. Then, it initializes internal variables. However, if an error occurs, no dialogs appear. Instead, an error is returned by CB.MacroResultDetail ("CB.MacroResultDetail" on page 176).

Call CB.CheckDataND or CB.CheckData each time you start a new enumeration, use a CB.Get... call that relies on cell references that might have changed, and each time you delete or add data. Use CB.CheckDataND or CB.CheckData before using calls that extract data or create reports.

CB.CheckDataND Example 1

This example runs a simulation of 500 trials, initializes all the internal variables for the following enumeration, and then enumerates all the forecasts, opening each forecast chart.

```
Dim s As String
Dim r As Range
CB.Simulation 500
CB.CheckDataND
s = CB.EnumFore
While s <> ""
   Set r = Range(s)
   CB.OpenFore
   s = CB.EnumFore
   t = t + 1
Wend
```

CB.CheckDataND Example 2

This example runs CB.CheckDataND, checks for errors, and displays a message if one occurs. Any following code runs if the error check is successful.

```
Sub ErrorHandling()

  ' Check the data

CB.CheckDataND

' Check if CheckDataND failed

If CB.MacroResult <> 0 Then

' Notify user or log error

MsgBox CB.MacroResultDetail.Msg

Exit Sub

End If

' Insert code that expects CheckDataND to have completed

End Sub
```

CB.CheckDataND Related Calls

Name	Description
CB.CheckData	Checks Crystal Ball data cells for validity, initializes variables with dialogs in case of error
CB.EnumAssum	Enumerates assumption cells for all open workbooks
CB.EnumFore	Enumerates forecast cells for all open workbooks
CB.EnumDecVar	Enumerates decision variable cells for all open workbooks

CB.ClearData

Subtopics

- CB.ClearData Example
- CB.ClearData Related Calls

This subroutine prompts you to confirm and then erases assumptions, decision variables, or forecasts from selected cells. Any values or formulas in the cells remain.

The worksheet must be active and the selected cell or range of cells must contain assumptions, decision variables, or forecasts. If the range contains more than one type of Crystal Ball data, a dialog appears and you can select the data types to clear.

Before you clear data, reset the current simulation.

Note: Before calling this subroutine, select a cell or cell range containing Crystal Ball data and reset the simulation.

CB.ClearData Example

The following lines of code reset the simulation, prompt the user to clear any Crystal Ball data in the currently selected cells, and then clear the data:

```
CB.ResetND
CB.ClearData
```

The cell contents, such as a value, cell reference, or formula, remain.

CB.ClearData Related Calls

Name	Description
CB.ClearDataND	Clears assumptions, decision variables, and forecasts from selected cells without confirmation
CB.CopyData	Copies Crystal Ball data in the specified range

Name	Description
CB.CopyDataND	Copies selected assumptions, decision variables, or forecasts
CB.PasteData	Pastes assumptions, decision variables, or forecasts into selected cells

CB.ClearDataND

Subtopics

- CB.ClearDataND Example
- CB.ClearDataND Related Calls

This subroutine erases assumptions, decision variables, and forecasts from selected cells. The values and formulas in the cells remain.

The worksheet must be active and the selected cell or range of cells must contain assumptions, decision variables, or forecasts. If the range contains more than one type of cell, all types are cleared.

Unlike CB.ClearData, CB.ClearDataND does not ask the user for confirmation.

Note: Before calling this subroutine, reset the current simulation and select a cell or cell range containing Crystal Ball data.

CB.ClearDataND Example

The following lines of code reset the simulation and clear the selected Crystal Ball data without any prompting:

```
CB.ResetND
CB.ClearDataND
```

The cell contents, such as a value, cell reference, or formula, remain.

CB.ClearDataND Related Calls

Name	Description
CB.ClearData	Erases assumptions, decision variables, and forecasts from selected cells after prompting
CB.CopyData	Copies Crystal Ball data in specified range
CB.CopyDataND	Copies selected assumptions, decision variables, or forecasts
CB.PasteData	Pastes assumptions, decision variables, or forecasts into selected cells

CB.CloseAllCharts

Subtopics

- CB.CloseAllCharts Example
- CB.CloseAllCharts Related Calls

This call closes all charts of any type. It is the equivalent of the Analyze, then Close All command. This call works on both current and restored results

CB.CloseAllCharts Example

This example runs a simulation and closes all open charts.

```
CB.Simulation 1000
CB.CloseAllCharts
```

CB.CloseAllCharts Related Calls

Name	Description
CB.CloseChart	Closes the selected chart of the given type
CB.OpenChart	Opens the selected chart of the specified type

CB.CloseChart

Subtopics

- CB.CloseChart Example
- CB.CloseChart Related Calls

This call closes a chart of the specified type and chart ID. The chart ID is optional. If the chart ID is not specified, the currently selected chart is closed. If the specified chart is not found, a chooser dialog appears. No error is returned.

Currently, this call only works on current and not restored results.

Table 25 CB.CloseChart Parameters

Parameter	VBA Data Type	Value	Description
SimChartType	Integer	cbChtOverlay = 1, cbChtScatter = 2, cbChtSensitiv = 3, cbChtTrend = 4	Specifies the type of chart you are closing: overlay, scatter, sensitivity, or trend chart. For named constant and index values, see the Value column.

Parameter	VBA Data Type	Value	Description
ChartID (Optional)	String	"*Chart ID*"	The chart ID returned by CB.EnumChart (Table 60) entered directly or using CB.EnumChart

CB.CloseChart Example

This example runs a simulation, calls CB.CheckData (because a CB.Enum call follows), and then closes the next overlay chart in the workbook.

```
CB.Simulation 1000
CB.CheckData
CB.CloseChart cbChtOverlay, CB.EnumChart(cbChtOverlay)
```

CB.CloseChart Related Calls

Name	Description
CB.CloseAllCharts	Closes all open chart windows; equivalent to Analyze, then Close All.
CB.EnumChart	Returns the chart ID of the next chart of the specified type
CB.OpenChart	Opens the selected chart of the specified type
CB.SelectChart	Selects the chart with the specified chart ID

CB.CloseFore

Subtopics

- CB.CloseFore Example
- CB.CloseFore Related Calls

This subroutine closes the forecast window for the selected cell.

The worksheet must be active and a forecast cell selected for Crystal Ball to run this command.

Note: Before calling this subroutine, activate a worksheet and select one forecast cell.

CB.CloseFore Example

This example activates the Model worksheet of the Futura With OptQuest.xls workbook and selects cell C9, a forecast. Then it closes the forecast window associated with that cell.

```
Workbooks("Futura with OptQuest.xls").Worksheets("Model").Activate
Range("C9").Select
CB.CloseFore
```

CB.CloseFore Related Calls

Name	Description
CB.OpenFore	Opens forecast window for selected cell

CB.CloseSensitiv

Subtopics

- CB.CloseSensitiv Example
- CB.CloseSensitiv Related Calls

This call closes the selected sensitivity chart. If more than one chart is found but one is not selected, a chooser dialog appears.

Note: This call is included for compatibility with Crystal Ball 2000.x (5.x). However, it differs somewhat from earlier versions. For details, see "CB.CloseSensitiv" on page 239). To close a chart of a specific type by its chart ID, use CB.CloseChart ("CB.CloseChart" on page 70).

CB.CloseSensitiv Example

This example runs a simulation, calls CB.CheckData (because a CB.Enum... call follows), selects the next sensitivity chart in the workbook and then closes it.

```
CB.Simulation 1000
CB.CheckData
CB.SelectChart cbChtSensitiv, CB.EnumChart(cbChtSensitiv)
CB.CloseSensitiv
```

CB.CloseSensitiv Related Calls

Name	Description
CB.CopySensitiv	Copies the selected sensitivity chart to the clipboard
CB.EnumChart	Returns the unique name of the next chart of the specified type
CB.OpenSensitiv	Opens the selected sensitivity chart
CB.SelectChart	Selects the chart with the specified chart ID

CB.CloseTrend

Subtopics

- CB.CloseTrend Example
- CB.CloseTrend Related Calls

This call closes the selected trend chart. If more than one chart is found but one is not selected, a chooser dialog appears.

Note: This call is included for compatibility with Crystal Ball 2000.x (5.x). However, it differs somewhat from earlier versions. For details, see "CB.CloseTrend" on page 239). To close a chart of a specific type by its chart ID, use CB.CloseChart ("CB.CloseChart" on page 70).

CB.CloseTrend Example

This example runs a simulation, calls CB.CheckData (because a CB.Enum... call follows), selects the next trend chart in the workbook and then closes it.

```
CB.Simulation 1000
CB.CheckData
CB.SelectChart cbChtTrend, CB.EnumChart(cbChtTrend)
CB.CloseTrend
```

CB.CloseTrend Related Calls

Name	Description
CB.EnumChart	Returns the unique name of the next chart of the specified type
CB.OpenTrend	Opens the selected trend chart
CB.SelectChart	Selects the chart with the specified chart ID

CB.CopyData

Subtopics

- CB.CopyData Example
- CB.CopyData Related Calls

This subroutine stores a selected range definition for later use by CB.PasteData, which pastes all current Crystal Ball data of the selected type or types from the range stored by CB.CopyData. You can use CB.CopyData with CB.PasteData to quickly set up rows or columns of forecasts, assumptions with similar distribution types, or decision variables with similar bounds. The

worksheet must be active, and the selected cell or range of cells must contain assumptions, decision variables, or forecasts.

Note: Before calling this subroutine, select a cell or cell range containing Crystal Ball data.

CB.CopyData Example

This example prompts you to select which data to copy — either the assumption or forecast data. It then copies only the selected data within the selected range.

```
'Range selected in the following line contains an assumption and a forecast
Range ("B5:C7").Select 'contains an assumption and a forecast
CB.CopyData
```

CB.CopyData Related Calls

Name	Description
CB.CopyDataND	Copies selected assumptions, decision variables, or forecasts
CB.PasteData	Pastes assumptions, decision variables, or forecasts into selected cells

CB.CopyDataND

Subtopics

This subroutine stores a range definition for later use by CB.PasteData, which pastes all current Crystal Ball data of the selected type from the range stored by CB.CopyDataND. You can use CB.CopyDataND with CB.PasteData to quickly set up rows or columns of forecasts, assumptions with similar distribution types, or decision variables with similar bounds. The worksheet must be active, and the selected cell or range of cells must contain assumptions, decision variables, or forecasts.

This call has one parameter, DataType, described in Table 26..

Table 26 CB.CopyDataND DataType Parameter — Integer, Required

Named Constant Value	Index Value	Description
cbCelAssum	1	Copies only assumption cell data.
cbCelFore	2	Copies only forecast cell data.
cbCelDecVar	3	Copies only decision variable cell data.

Use this subroutine when the data range to copy has multiple Crystal Ball data types within the range.

This subroutine, compared to CB.CopyData, pastes only the specified data type from a range with mixed cells. If a range contains different Crystal Ball data types (assumptions, decision variables, and forecasts), then Crystal Ball will only copy the data type you specify with the *DataType* parameter.

Note: Before calling this subroutine, select a cell or cell range containing Crystal Ball data.

CB.CopyDataND Example

This example copies only the Crystal Ball assumption data from cells B5 and B6.

```
Range("B5:B6").Select 'contains an assumption and a forecast
CB.CopyDataND cbCelAssum
```

CB.CopyDataND Related Calls

Name	Description
CB.CopyData	Copies Crystal Ball data in specified range
CB.PasteData	Pastes assumptions, decision variables, or forecasts into selected cells

CB.CopyScatter

Subtopics

This call copies the selected scatter chart to the clipboard. If more than one chart is found but one is not selected, a chooser dialog appears.

CB.CopyScatter Example

This example runs a simulation, calls CB.CheckData (because a CB.Enum... call follows), selects the next scatter chart in the workbook and then copies it.

```
CB.Simulation 1000
CB.CheckData
CB.SelectChart cbChtScatter, CB.EnumChart(cbChtScatter)
CB.CopyScatter
```

CB.CopyScatter Related Calls

Name	Description
CB.CloseChart	Closes the selected chart of the given type
CB.EnumChart	Returns the unique name of the next chart of the specified type
CB.OpenChart	Opens the selected chart of the specified type
CB.SelectChart	Selects the chart with the specified chart ID

CB.CopySensitiv

Subtopics

- CB.CopySensitiv Example
- CB.CopySensitiv Related Calls

This call copies the selected sensitivity chart to the clipboard. If more than one chart is found but one is not selected, a chooser dialog appears.

CB.CopySensitiv Example

This example runs a simulation, calls CB.CheckData (because a CB.Enum... call follows), selects the next sensitivity chart in the workbook and then copies it.

```
CB.Simulation 1000
CB.CheckData
CB.SelectChart cbChtSensitiv, CB.EnumChart(cbChtSensitiv)
CB.CopySensitiv
```

CB.CopySensitiv Related Calls

Name	Description
CB.CloseSensitiv	Closes the selected sensitivity chart
CB.EnumChart	Returns the unique name of the next chart of the specified type
CB.OpenSensitiv	Opens the selected sensitivity chart
CB.SelectChart	Selects the chart with the specified chart ID

CB.CopyTrend

This call copies the selected trend chart to the clipboard. If more than one chart is found but one is not selected, a chooser dialog appears.

CB.CopyTrend Example

This example runs a simulation, calls CB.CheckData (because a CB.Enum... call follows), selects the next trend chart in the workbook and then copies it.

```
CB.Simulation 1000
CB.CheckData
CB.SelectChart cbChtTrend, CB.EnumChart(cbChtTrend)
CB.CopyTrend
```

CB.CopyTrend Related Calls

Name	Description
CB.CloseTrend	Closes the selected trend chart
CB.EnumChart	Returns the unique name of the next chart of the specified type
CB.OpenTrend	Opens the selected trend chart
CB.SelectChart	Selects the chart with the specified chart ID

CB.CorrelateND

This subroutine defines or updates a correlation coefficient between the assumptions corresponding to two selected cells.

Table 27 CB.CorrelateND Parameters

Parameter	VBA Data Type	Value	Description
Coefficient	Variant	A number between –1.0 and 1.0.	Sets the correlation coefficient between two selected assumptions
SecondAsm (Optional)	Microsoft Excel. Range	A single cell outside of the specified range on the same or another worksheet	Sets a single assumption cell as the second cell to correlate with the first specified cell in a selected range. The second cell specified with SecondAsm should not be part of the selected range and can be on another worksheet in the same workbook.

To specify two assumption cells to correlate, either:

- Select a rectangular cell range so that the two assumptions define opposite corners or endpoints of a column or row, or

- Select a single assumption cell and use the optional SecondAsm parameter to define a second assumption cell on the same worksheet or another worksheet on the same workbook.

If the cells to correlate are in the lower left (the lowest column and the highest row) and upper right (highest column and lowest row) corners of the selected range, activate one of the cells before calling this subroutine. If the two assumption cells are in the same column or row, the selected cell range is one cell wide or tall. If the cells are adjacent, the selected region has exactly two cells in it.

Unless you are using the SecondAsm parameter, both the selected cell and the opposite corner of the selected range must contain assumption cells. If the selected region is just one cell, use SecondAsm to define a second cell, because you cannot correlate a cell with itself. If there is an error in specifying the cells to be correlated, CB.CorrelateND returns cbErrNoAssumptionInCell. If you enter a number outside the range -1.0 and 1.0, the subroutine deletes any correlation between the cells.

See the *Oracle Crystal Ball User's Guide* for more information on specifying correlations between assumptions.

CB.CorrelateND Example 1

This example correlates cell B5 with F12, with a correlation coefficient of 0.2.

```
Range("B5:F12").Select
CB.CorrelateND 0.2
```

CB.CorrelateND Example 2

This example correlates cell B12 with F5, with a correlation coefficient of 0.6.

```
Range("F5:B12").Select
Range("B12").Activate
CB.CorrelateND 0.6
```

CB.CorrelateND Example 3

This example correlates cells B5 and B12 with a correlation coefficient of 0.3.

```
Range("B5:B12").Select
CB.CorrelateND 0.3
```

CB.CorrelateND Related Calls

Name	Description
CB.DefineAssum	Defines or changes assumptions in selected cells using a dialog
CB.DefineAssumND	Defines or changes assumptions in selected cells without using a dialog

CB.CreateChart

This function creates a new chart of the specified type and returns the chart ID of the new chart. Optionally, you can specify the name. If successful, the new chart becomes the selected chart for subsequent chart calls. If the chart is not created successfully, an empty string is returned.

Table 28 CB.CreateChart Returned Data Type

Returned Value	Returned Data Type
The path and name of the new chart of the specified type, ChartID, in the format: "|Current Results|Sensitivity Charts|Charts.xls|Sensitivity Chart 1"	String

Table 29 CB.CreateChart Parameters and Constants

Parameter	VBA Data Type	Value	Description
SimChartType	Integer	cbChtOverlay = 1, cbChtScatter = 2, cbChtSensitiv = 3, cbChtTrend = 4	Specifies the type of chart you are creating: overlay, scatter, sensitivity, or trend chart. For named constant and index values, see the Value column.
DisplayName (Optional)	String	"*Chart ID*"	User-defined or default chart name (title) entered directly

CB.CreateChart Example

This example creates an overlay chart, a trend chart, and a sensitivity chart and names each with the names in quotes.

```
ChartID = CB.CreateChart(cbChtOverlay, "Overlay Chart 1")
ChartID = CB.CreateChart(cbChtTrend, "Trend Chart 1")
ChartID = CB.CreateChart(cbChtSensitiv, "Sensitivity Chart 1")
```

CB.CreateChart Related Calls

Name	Description
CB.CheckData	Checks decision variables for validity, initializes variables
CB.EnumChart	Returns the unique name of the next chart of the specified type
CB.SelectChart	Selects the chart with the specified chart ID

CB.CreateRpt

This subroutine creates a Crystal Ball report. It opens the Create Report dialog that enables you to select report options and create a report. Use CB.CheckData before using this call.

CB.CreateRpt Example

This example opens the Create Report dialog, so you can include assumptions, decision variables, forecasts, and charts.

```
CB.Simulation 1000
CB.CheckData
CB.CreateRpt
```

CB.CreateRpt Related Calls

Name	Description
CB.CreateRptND	Sets report preferences and creates a report from the current simulation and restored results, optimization, or time-series forecast without a dialog
CB.ExtractData	Extracts data from the current simulation and restored results, optimization, or time-series forecast using a dialog
CB.ExtractDataND	Extracts data from the current simulation and restored results, optimization, or time-series forecast without a dialog

CB.CreateRptND

This subroutine sets report preferences and creates a Crystal Ball report. For any preferences not manually set, the default is the current setting in the Create Report dialog. Use CB.CheckData before using this call.

Note: Although Full is the default report type in the Create Report dialog, Custom (cbRptCustom) is the default report for CB.CreateRptND.

To create a custom report, call CB.CreateRptND as follows before creating the report:

```
CB.CreateRptND cbRptDefinedType, cbRptCustom
```

Table 30 CB.CreateRptND Parameters

Parameter	VBA Data Type	Description
Index	Integer	Specifies which report preference you are setting. See Table 31.
Value1 (Optional)	Variant	Specifies settings for the preference indicated by Index. See Table 31.
Value2 (Optional)	Variant	Specifies settings for the preference indicated by Index. See Table 31.

Table 31 shows each *Index* value and the corresponding *Value1* and *Value2* parameter values.

Table 31 Index and corresponding Value1 and Value2 Parameters

Named Constant Value	Index Value	Description	Value1 (Optional)	Value2 (Optional)
cbRptOK	1	Generates the report. Use this parameter after setting other report options.	n/a	n/a
cbRptTrendChart	2	Includes all open trend charts in the report at a specified scale	(Boolean) True includes the chart; False does not.	(Integer) 1 to 200 defines the percent scale of the chart.
cbRptSensitivChart	3	Includes all open sensitivity charts in the report at a specified scale	(Boolean) True includes the chart; False does not.	(Integer) 1 to 200 defines the percent scale of the chart.
cbRptForeSummaries	4	Includes summary information about the included forecasts	(Boolean) True includes the summaries. False does not.	n/a
cbRptForeStatistics	5	Includes the statistics table for each included forecast	(Boolean) True includes the statistics. False does not.	n/a
cbRptForeCharts	6	Includes the frequency chart for each included forecast at a specified scale	(Boolean) True includes the chart; False does not.	(Integer) 1 to 200 defines the percent scale of the chart.
cbRptForePercentiles	7	Includes the percentile statistics for each included forecast	(Boolean) True includes the percentiles. False does not.	(Integer) See Table 32.
cbRptChooseFore	9	enables you to clear, add, and set the list of forecasts to include in the report	(Integer) See table Table 33, following.	n/a
cbRptAssumParam	10	Includes the distribution parameters for each included assumption	(Boolean) True includes the parameters. False does not.	n/a
cbRptAssumCharts	11	Includes the distribution graph for each included assumption at the specified scale	(Boolean) True includes the chart; False does not.	(Integer) 1 to 200 defines the percent scale of the chart.
cbRptChooseAssum	12	Clears, adds, and sets the assumptions to include in the report	(Integer) See table Table 34 following.	n/a
cbRptChartType	13	Selects the type of chart to include in the report	(Integer) See Table 35 following.	n/a
cbRptOverlayChart	14	Includes all open overlay charts in the report at a specified scale	(Boolean) True includes the chart; False does not.	(Integer) 1 to 200 defines the percent scale of the chart.

Named Constant Value	Index Value	Description	Value1 (Optional)	Value2 (Optional)
cbRptExistingSheet	16	Creates the report in an existing worksheet	(Boolean) True creates the report in a new page of the existing workbook. False creates the report in a new workbook.	n/a
cbRptSummary	17	Specifies which summary information to include in the report	(Boolean) True includes the parameter specified in Value2; False does not.	(Integer) See Table 36.
cbRptChooseDecVar	18	Clears, adds, and sets the list of decision variables to include in the report	(Integer) See Table 37.	n/a
cbRptDecVarBounds	19	Includes the bounds for each included decision variable	(Boolean) True includes the bounds; False does not.	n/a
cbRptDecVarType	20	Includes the type for each included decision variable	(Boolean) True includes the type; False does not.	n/a
cbRptAssumStatistics	21	Includes the statistics table for each included assumption	(Boolean) True includes the statistics; False does not.	n/a
cbRptAssumPercentiles	22	Includes the percentile data for each included assumption	(Boolean) True includes percentile data; False does not.	(Integer) Same as that for cbRptFore-Percentiles. See Table 32 following.
cbRptAssumCorr	23	Includes the correlations for each included assumption	(Boolean) True includes the correlations; False does not.	n/a
cbRptOverlayCharts	24	Includes the distribution graph for each included overlay chart	(Boolean) True includes the charts; False does not.	(Integer) 1 to 200 defines the percent scale of the charts.
cbRptTrendCharts	25	Includes the graph for each included trend chart	(Boolean) True includes the charts; False does not.	(Integer) 1 to 200 defines the percent scale of the charts.
cbRptSensitivityCharts	26	Includes the graph for each included sensitivity chart	(Boolean) True includes the charts; False does not.	(Integer) 1 to 200 defines the percent scale of the charts.
cbRptScatterCharts	27	Includes the graph for each included scatter chart	(Boolean) True includes the charts; False does not.	(Integer) 1 to 200 defines the percent scale of the charts.
cbRptChooseOverlay	28	Clears, adds, and sets the overlay charts to include in the report (when used with CB.SelectChart)	(Integer) See Table 38. If cbChartAdd, the selected chart is added.	n/a
cbRptChooseTrend	29	Clears, adds, and sets the trend charts to include in the report (when used with CB.SelectChart)	(Integer) See Table 38. If cbChartAdd, the selected chart is added.	n/a

Named Constant Value	Index Value	Description	Value1 (Optional)	Value2 (Optional)
cbRptChooseSensitivity	30	Clears, adds, and sets the sensitivity charts to include in the report (when used with CB.SelectChart)	(Integer) See Table 38. If cbChartAdd, the selected chart is added.	n/a
cbRptChooseScatter	31	Clears, adds, and sets the scatter charts to include in the report (when used with CB.SelectChart)	(Integer) See Table 38. If cbChartAdd, the selected chart is added.	n/a
cbRptDefinedType	32	Selects a predefined report type or uses the settings defined by the various CB. CreateRptND constants	(Integer) See Table 39.	n/a
cbRptSheetName	33	Specifies the sheet name	(String) The sheet name to create.	n/a
cbRptIncludeCellLocs	34	Includes the cell location in the report	(Boolean) True includes the cell location information; False does not.	n/a
cbRptSection	35	Indicates whether to include or exclude entire report sections	(Boolean) True includes the report section specified in Value2; False excludes the specified section.	(Integer) See Table 40.
cbRptSectionOrder	36	Specifies the order of the report sections. **Note:** Sections must be explicitly included using cbRptSection statements.	(String) A string of ten comma-separated values (listed in Table 40) that specify the section order. For example: 1,6,7,8,2,3,4,5,9, 10 indicates that the report should include sections in this order: Summary, Forecasts, Assumptions, Decision Variables, Overlay, Trend, Sensitivity, Scatter, OptQuest results, Predictor forecasts	n/a
cbRptCapMetrics	37	Specifies whether to report capability metrics for forecasts in custom reports	True includes the metrics; False does not include the metrics.	n/a
cbRptScatterChart	38	Includes all open scatter charts in the report at a specified scale	(Boolean) True includes the chart; False does not.	(Integer) 1 to 200 defines the percent scale of the chart.
cbRptOptimizerSummary	39	Includes the OptQuest Summary section in the report following an optimization run	(Boolean) True includes the OptQuest summary information; False does not.	n/a
cbRptOptimizerChart	40	Includes the optimization solution graph in the report following an optimization run	(Boolean) True includes the chart; False does not.	(Integer) 1 to 200 defines the percent scale of the chart.

Named Constant Value	Index Value	Description	Value1 (Optional)	Value2 (Optional)
cbRptOptimizerBestSolution	41	Includes the OptQuest Best Solution section in the report following an optimization run	(Boolean) True includes the OptQuest best solution information; False does not.	n/a
cbRptOptimizerConstraints	42	Includes the OptQuest Constraints section in the report following an optimization run	(Boolean) True includes the OptQuest constraints information; False does not.	n/a
cbRptOptimizerDecVars	43	Includes the OptQuest Decision Variables section in the report following an optimization run	(Boolean) True includes the OptQuest decision variables information; False does not.	n/a
cbRptPredictorChart	45	Includes Predictor results charts for each series in the report following a Predictor run.	(Boolean) True includes the charts; False does not.	(Integer) 1 to 200 defines the percent scale of the charts.
cbRptPredictorStatistics	46	Includes the Predictor Statistics section in the report following a Predictor run	(Boolean) True includes the Predictor statistics information; False does not.	n/a
cbRptPredictorAutocorrelations	47	Includes the Predictor Autocorrelations section in the report following a Predictor run	(Boolean) True includes the Predictor autocorrelations information; False does not.	n/a
cbRptPredictorForecast	48	Includes the Predictor Forecasts section in the report following a Predictor run	(Boolean) True includes the Predictor forecasts information; False does not.	n/a
cbRptPredictorConfidenceIntervals	49	Includes the Predictor Confidence Intervals section in the report following a Predictor run	(Boolean) True includes the Predictor confidence intervals information; False does not.	n/a
cbRptPredictorMethods	50	Indicates which Predictor Methods to include in the report	(Integer) See Table 41 following.	n/a

Note: The following constants have been removed since Crystal Ball 2000.5 (5.5): cbRptForeFrequencies = Index 8 (obsolete), cbRptDecVar = Index 15 (included in CB 7.1 and later for compatibility; users should migrate to using cbRptChooseDecVar instead). cbRptScatterChart [38] and similar constants such as cbRptOverlayChart [14] include all open charts of the selected type, as stated in their descriptions. These constants override any previous settings made with "Choose" commands, such as cbRptChooseScatter [31]. The cbRptOptimizer... constants (39-43) were added in Crystal Ball 11.1.1.0.00

With *Index* set to cbRptForePercentiles, you can specify which percentile ranges to include in the report (see Table 32).

Table 32 For Index = cbRptForePercentiles, Value2 Values — Optional, Integer

Named Constant Value	Index Value	Percentile Ranges
cbPctQuartiles	1	Quartiles (25%)
cbPctQuintiles	2	Quintiles (20%).
cbPctDeciles	3	Deciles (10%). This is the default.
cbPctIcosatiles	4	Icosatiles (5%)
cbPctSet1	5	2.5, 5, 50, 95, 97.5%-iles
cbPctSet2	6	10, 25, 50, 75, 90%-iles
cbPctSet3	7	5, 95% percentiles
cbPctSet4	8	10, 90% percentiles

With *Index* set to cbRptChooseFore, you can specify which forecasts to include in the report (see Table 33).

Table 33 For Index = cbRptChooseFore, Value1 Values — Optional, Integer

Named Constant Value	Index Value	Description
cbChfAll	1	Includes all forecasts
cbChfOpen	2	Includes open forecasts
cbChfChosen	3	Includes chosen forecasts. Run this option after adding forecasts.
cbChfAdd	4	Adds the forecast in the selected cell to the list of chosen forecasts
cbChfClearList	5	Clears the list of chosen forecasts

With *Index* set to cbRptChooseAssum, you can specify which assumptions to include in the report (see Table 34).

Table 34 For Index = cbRptChooseAssum, Value1 Values — Optional, Integer

Named Constant Value	Index Value	Description
cbChaAll	1	Includes all assumptions
cbChaChosen	2	Includes chosen assumptions. Run this option after adding assumptions.
cbChaAdd	3	Adds the assumption in the selected cell to the chosen assumptions. Use cbChaChosen after this option.
cbChaClearList	4	Clears the list of chosen assumptions

Named Constant Value	Index Value	Description
cbChaOpen	5	Includes all open assumptions

With *Index* set to cbRptChartType, you can specify what type of chart the report uses (see Table 35).

Table 35 For Index = cbRptChartType, Value1 Values — Optional, Integer

Named Constant Value	Index Value	Description
cbChtColored	1	Makes all charts colored, Crystal Ball format
cbChtExcel	2	Makes all charts Microsoft Excel format

Note: cbChtBW = 0 was removed as an index to this parameter.

With *Index* set to cbRptSummary, you can specify what to include in the summary (see Table 36).

Table 36 For Index = cbRptSummary, Value2 Values — Optional, Integer

Named Constant Value	Index Value	Description
cbRptSummaryTitle	1	Includes the report title
cbRptSummaryDate	2	Includes the date
cbRptRunPreferences	3	Includes run preferences for the simulation
cbRptRunStatistics	4	Includes run statistics for the simulation

With *Index* set to cbRptChooseDecVar, you can specify which decision variables to include in a report (see Table 37).

Table 37 Index = cbRptChooseDecVar, Value1 Values — Optional, Integer

Named Constant Value	Index Value	Description
cbChdAll	1	Includes all decision variables
cbChdChosen	2	Includes chosen decision variables. Include a statement with this constant after added decision variables
cbChdAdd	3	Adds the decision variable in the selected cell to the list of chosen decision variables
cbChdClearList	4	Clears the list of chosen decision variables

With *Index* set to cbRptChooseOverlay, cbRptChooseTrend, or cbRptChooseSensitivity, you can specify which individual charts to include in a report when charts are selected with CB.SelectChart (see Table 38).

Table 38 For Index = cbRptChooseChartype, Value1 Values — Optional, Integer

Named Constant Value	Index Value	Description
cbChartAll	1	Includes all charts of the specified type
cbChartOpen	2	Includes all open charts of the specified type
cbChartChosen	3	Includes chosen charts of the specified type
cbChartAdd	4	Adds the selected chart to the list of chosen charts
cbChartClearList	5	Clears the list of chosen charts

With *Index* set to cbRptDefinedType, you can indicate which predefined report type to use (see Table 39).

Table 39 For Index = cbRptDefinedType, Value1 Values — Optional, Integer

Named Constant Value	Index Value	Description
cbRptFull	1	Report format is the predefined full report
cbRptAssumption	2	Report format is the predefined assumption report
cbRptForecasts	3	Report format is the predefined forecast report
cbRptDecVars	4	Report format is the predefined decision variable report
cbRptIndex	5	Report format is the predefined index report
cbRptCustom	6	Report format is set through various CB.CreateRptND code statements
cbRptOptimizer	7	Report format is the predefined OptQuest report, available following an OptQuest optimization run
cbRptPredictor	8	Report format is the predefined Predictor report, available following a Predictor forecasting run

With *Index* set to cbRptSection, you can use the following constants as the *Value2* parameter to specify the order of report sections (see Table 40).

Table 40 For Index = cbRptSection, Value2 Values — Optional, Integer

Named Constant Value	Index Value	Description
cbRptSectSummary	1	Includes the summary section
cbRptSectOverlay	2	Includes the overlay chart section
cbRptSectTrend	3	Includes the trend chart section
cbRptSectSensitivity	4	Includes the sensitivity chart section
cbRptSectScatter	5	Includes the scatter chart section
cbRptSectForecasts	6	Includes the forecasts section

Named Constant Value	Index Value	Description
cbRptSectAssumptions	7	Includes the assumptions section
cbRptSectDecVars	8	Includes the decision variables section
cbRptSectOptimizerResults	9	Includes the OptQuest results section, following an OptQuest optimization run
cbRptSectPredictorSeries	10	Includes the Predictor results section, following a Predictor forecasting run

With *Index* set to cbRptPredictorMethods, you can use the following constants as the *Value1* parameter to specify which forecasting methods to include in Predictor report data (see Table 41).

Table 41 For Index = cbRptPredictorMethods, Value1 Values — Optional, Integer

Named Constant Value	Index Value	Description
cbPredictorMethodsNone	1	Includes no methods
cbPredictorMethodsAll	2	Includes all methods
cbPredictorMethods1Best	3	Includes the method with the best accuracy
cbPredictorMethods2Best	4	Includes the method with the second-best accuracy
cbPredictorMethods3Best	5	Includes the method with the third-best accuracy

Use cbRptOK to generate the report after you use the other index values to set report preferences.

A simulation must not be in progress. If a simulation has not yet run, the report only includes assumption information.

Note: To add individual charts, use constants cbRptChooseOverlay, cbRptChooseTrend, and cbRptChooseSensitivity from Table 30. Use these constants with CB.SelectChart and the constants in Table 38. If you create a chart through the Developer Kit, open it before including it in a report. Otherwise, the chart will show up in the report with a "no data available" message. See the *Oracle Crystal Ball User's Guide* for more information on creating reports.

CB.CreateRptND Example 1

This example runs a simulation (so that the report can have forecast and charts in it). Then it defines the report type as Custom, sets preferences not to include a trend chart, an overlay chart, or any assumption information. It does include forecast charts for all the forecasts, but no other forecast statistics or information. The charts are in Microsoft Excel format.

When you use a ClearList constant, it clears all individually selected items of that type. However, if you have used an All constant, such as cbChfAll, using cbChfClearList is overridden by the All constant and, in this case, information is still included for all forecasts. For this reason, it is good practice to use a Chosen constant after a ClearList constant.

```
CB.Simulation 1000
CB.CheckData
CB.CreateRptND cbRptSection, True, cbRptSectForecasts
CB.CreateRptND cbRptDefinedType, cbRptCustom
CB.CreateRptND cbRptTrendCharts, False
CB.CreateRptND cbRptSensitivityCharts, True, 150
CB.CreateRptND cbRptOverlayCharts, False
CB.CreateRptND cbRptChooseFore, cbChfAll
CB.CreateRptND cbRptForeSummaries, False
CB.CreateRptND cbRptForeStatistics, False
CB.CreateRptND cbRptForeCharts, True, 50
CB.CreateRptND cbRptForePercentiles, True, cbPctDeciles
CB.CreateRptND cbRptChooseAssum, cbChaClearList
CB.CreateRptND cbRptChooseAssum, cbChaChosen
CB.CreateRptND cbRptChartType, cbChtExcel
CB.CreateRptND cbRptOK
```

CB.CreateRptND Example 2

This example resets any previously run simulation, then clears the current assumption list, adds B5 to the assumption list, and turns on the assumption parameter option (the second cbChaChosen use). Other assumption preferences default to the current setting in the Create Report dialog. Finally, the example generates the report. No forecast or simulation charts will appear, since CB.ResetND cleared all previous simulation information.

```
CB.ResetND
CB.CheckData
CB.CreateRptND cbRptChooseAssum, cbChaClearList
CB.CreateRptND cbRptChooseAssum, cbChaChosen
Range("B5").Select
CB.CreateRptND cbRptChooseAssum, cbChaAdd
CB.CreateRptND cbRptChooseAssum, cbChaChosen
Cb.CreateRptND cbRptAssumParam, True
CB.CreateRptND cbRptOK
```

CB.CreateRptND Related Calls

Name	Description
CB.CreateRpt	Sets report preferences and creates a report from the current simulation and restored results, optimization, or time-series forecast using a dialog
CB.ExtractData	Extracts data from the current simulation and restored results, optimization, or time-series forecast using a dialog
CB.ExtractDataND	Extracts data from the current simulation and restored results, optimization, or time-series forecast without a dialog

CB.DataAnalysis

This call launches the Data Analysis tool wizard. This tool helps you import data directly into Crystal Ball forecasts for further analysis.

Note: Before calling CB.DataAnalysis, reset the simulation.

CB.DataAnalysis Example

This example resets the Crystal Ball simulation and opens the Data Analysis wizard.

```
CB.ResetND
CB.DataAnalysis
```

CB.DataAnalysis Related Calls

Name	Description
CB.GetDataAnalysisOption	Returns current settings for the Data Analysis tool
CB.DataAnalysisND	Imports data into Crystal Ball forecasts without using a dialog

CB.DataAnalysisND

The Data Analysis tool imports data directly into Crystal Ball forecasts for further analysis. This call runs the Data Analysis tool and displays results without using a dialog.

Note: Before calling CB.DataAnalysisND, reset the Crystal Ball simulation.

Table 42 CB.DataAnalysisND Parameters

Parameter	Required?	VBA Data Type	Table
Index	Required	Integer	Table 43
Value	Optional	Variant	Table 44

Table 43 CB.DataAnalysisND Index Parameter — Required, Integer

Named Constant Value	VBA Data Type	Index Value	Description
cbDataAnalysisRun	Integer	1	Runs the Data Analysis tool when all settings are complete
cbDataAnalysisSelectData	Integer	2	Selects the numeric portion of the data to be imported. The range can either be specified as a separate line preceding the CB.DataAnalysisND call, such as [A1:C100].Select, or it can be the *Value* parameter, specified as "A1:C100" or similar.
cbDataAnalysisDataOrientation	Integer	3	Works with *Value* to specify whether data is in rows or columns: cbDataAnalysisDataInRows [1] cbDataAnalysisDataInColumns [2]

Named Constant Value	VBA Data Type	Index Value	Description
cbDataAnalysisSelectHeaders	Integer	4	Uses a boolean value for the *Value* parameter to include column header text in the selection; True = include the top row of text
cbDataAnalysisSelectLabels	Integer	5	Uses a boolean value for the *Value* parameter to include row label text in the selection; True = include the first column of text
cbDataAnalysisSelectLSL	Integer	6	Sets the selected range of values as forecast LSLs (lower specification limit) for process capability analysis. The range can either be specified as a separate line preceding the CB.DataAnalysisND call, such as [A1:C10].Select, or it can be the *Value* parameter, specified as "A1:C10" or similar.
cbDataAnalysisSelectUSL	Integer	7	Sets the selected range of values as forecast USLs (upper specification limit) for process capability analysis. See cbDataAnalysisSelectLSL for selection examples.
cbDataAnalysisSelectTarget	Integer	8	Sets the selected range of values as forecast Targets (target values) for process capability analysis. See cbDataAnalysisSelectLSL for selection examples.
cbDataAnalysisAutoOpenCharts	Integer	9	Uses a boolean value for the *Value* parameter to indicate whether to open forecast charts automatically when the data analysis runs; True = automatically open charts.
cbDataAnalysisSetViewType	Integer	10	If charts are shown, takes cbChtChartType constants from CB.ChartPrefsND as the *Value* parameter to determine which chart view appears. See Table 24 for constants.
cbDataAnalysisFitToDistribution	Integer	11	Uses a boolean value for the *Value* parameter to indicate whether forecast charts should be fitted to a distribution; True = fit to distributions.
cbDataAnalysisRankingMethod	Integer	12	If cbDataAnalysisFitToDistribution is True, works with *Value* to specify a ranking method for distribution fitting. See Table 78 for constants.
cbDataAnalysisGenerateCorrelationMatrix	Integer	13	Uses a boolean value for the *Value* parameter to indicate whether rank correlations between forecast charts should be displayed in a matrix; True = generate matrix.
cbDataAnalysisRunOnAllModels	Integer	14	Uses a boolean value for the *Value* parameter to indicate whether a simulation is run on all open workbooks at the same time the selected data is analyzed. In this case, forecast charts for all opened models appear with those generated from the data selected for analysis. True = run the simulation on all open workbooks; this setting can be used for validating models.
cbDataAnalysisResetSessionVariable	Integer	17	Resets all session variables related to the CB.DataAnalysisND Developer Kit call including the input data series and the LSL, USL, and Target capability values.

Table 44 CB.DataAnalysisND Value Parameter — Optional, Variant

Related Value	Named Constant Value	VBA Data Type	Index Value	Description
For *Index* = 2, cbDataAnalysisSelectData	A cell reference with format:Range("A1:C300") or "A1:C300"	Variant	n/a	Specifies the numeric portion of the data to be imported
For *Index* = 3, cbDataAnalysisDataOrientation	cbDataAnalysisDataInRows	Integer	1	Indicates that data cells are in rows
For *Index* = 3, cbDataAnalysisDataOrientation	cbDataAnalysisDataInColumns	Integer	2	Indicates that data cells are in columns
For *Index* = 4, cbDataAnalysisSelectHeaders	True or False	Boolean	n/a	Indicates whether to include column header text in the selection; True = include the top row of text
For *Index* = 5, cbDataAnalysisSelectLabels	True or False	Boolean	n/a	Indicates whether to include row label text in the selection; True = include the first column of text
For *Index* = 6, cbDataAnalysisSelectLSL	A cell reference with format:Range("A1:C300") or "A1:C300"	Variant	n/a	Sets the selected range of values as forecast LSLs (lower specification limit) for process capability analysis
For *Index* = 7, cbDataAnalysisSelectUSL	A cell reference with format:Range("A1:C300") or "A1:C300"	Variant	n/a	Sets the selected range of values as forecast USLs (upper specification limit) for process capability analysis
For *Index* = 8, cbDataAnalysisSelect Target	A cell reference with format:Range("A1:C300") or "A1:C300"	Variant	n/a	Sets the selected range of values as forecast Targets (target values) for process capability analysis
For *Index* = 9, cbDataAnalysisAutoOpenCharts	True or False	Boolean	n/a	Indicates whether to open forecast charts automatically when the data analysis runs; True = automatically open charts
For *Index* = 10, cbDataAnalysisSetViewType	Constants (Table 24)	Integer	See Table 24	Uses cbChtChartType constants from CB.ChartPrefsND to determine which chart view appears. See Table 24 for constants. The following views are allowed: frequency, cumulative frequency, reverse cumulative frequency, capability metrics, statistics, percentiles, and goodness of fit.

Related Value	Named Constant Value	VBA Data Type	Index Value	Description
For *Index* = 11, cbDataAnalysisFitTo Distribution	True or False	Boolean	n/a	Indicates whether forecast charts should be fitted to a distribution; True = fit to distributions
For *Index* = 12, cbDataAnalysisRankingMethod	Constants (Table 78)	Integer	See Table 78	Specifies a ranking method for distribution fitting. See Table 78 for constants
For *Index* = 13, cbDataAnalysisGenerate CorrelationMatrix	True or False	Boolean	n/a	Indicates whether rank correlations between forecast charts should be displayed in a matrix; True = generate matrix
For *Index* = 14, cbDataAnalysisRunOnAllModels	True or False	Boolean	n/a	Indicates whether a simulation is run on all open workbooks at the same time the selected data is analyzed; True = run the simulation on all open workbooks

CB.DataAnalysisND Example

This example resets the Crystal Ball simulation and fits all distributions to a columnar data set in cells B1 through D100 (including a header row). Forecast charts automatically appear when the simulation runs. They are fitted to a distribution. The selected ranking method is Chi-square. The last line of code runs the Data Analysis tool.

```
CB.ResetND

' Select data in cells B1 through D100
CB.DataAnalysisND cbDataAnalysisSelectData, "B1:D100"

' Set input range orientation (data in columns)
CB.DataAnalysisND cbDataAnalysisDataOrientation, cbDataAnalysisDataInColumns

' Input range has headers
CB.DataAnalysisND cbDataAnalysisSelectHeaders, True

' Automatically open forecast charts
CB.DataAnalysisND cbDataAnalysisAutoOpenCharts, True

' Fit to distribution
CB.DataAnalysisND cbDataAnalysisFitToDistribution, True

' Select the Chi-square ranking method
CB.DataAnalysisND cbDataAnalysisRankingMethod, cbFitChiSquare

' Run tool
CB.DataAnalysisND cbDataAnalysisRun
```

CB.DataAnalysisND Related Calls

Name	Description
CB.DataAnalysis	Launches the Data Analysis tool
CB.GetDataAnalysisOption	Returns current settings for the Data Analysis tool

CB.DecisionTable

This call launches the Decision Table tool wizard. This tool runs multiple simulations to test different values for one or two decision variables. The tool tests values across the range of the decision variables and puts the results in a table that you can analyze using Crystal Ball forecast, trend, or overlay charts.

Note: Before calling CB.DecisionTable, reset the simulation.

CB.DecisionTable Example

This example resets Crystal Ball and opens the Decision Table wizard.

```
CB.Reset
CB.DecisionTable
```

CB.DecisionTable Related Calls

Name	Description
CB.DecisionTableND	Runs the Decision Table tool without displaying dialogs
CB.GetDecisionTableOption	Returns current settings for the Decision Table tool

CB.DecisionTableND

The Decision Table tool runs multiple simulations to test different values for one or two decision variables. This call runs the Decision Table tool and displays results without using a dialog.

Note: Before calling CB.DecisionTableND, reset the Crystal Ball simulation.

Table 45 CB.DecisionTableND Parameters

Parameter	Required?	VBA Data Type	Table
Index	Required	Integer	Table 46

Parameter	Required?	VBA Data Type	Table
Index2	Optional	Integer	Table 47
Value	Optional	Variant	Table 48

Table 46 CB.DecisionTableND Index Parameter – Required, Integer

Named Constant Value	Index Value	Description
cbDecChooseTarget	1	Sets the target forecast for the decision table (operates on the forecast in the currently selected cell)
cbDecChooseDecVar	2	Used with *Index2* to select one or two decision variables for the decision table (operates on the decision variable in the currently selected cell)
cbDecSimOption	3	Used with *Index2* to set simulation options for the decision table
cbDecForeOption	4	Used with *Index2* to set forecast display options
cbDecRun	5	Runs the Decision Table tool when all settings have been specified

Table 47 CB.DecisionTableND Index2 Parameter – Optional, Integer

Related Value	Named Constant Value	Index Value	Description
For Index = 2, cbDec ChooseDecVar: Integer	cbChdAdd	3	Used with cbDecChooseDecVar to add the decision variable in the selected cell to the list of chosen decision variables
For Index = 2, cbDec ChooseDecVar: Integer	cbChdClearList	4	Used with cbDecChooseDecVar to clear the list of chosen decision variables
For Index = 3, cbDecSimOption: Integer	cbDecNumValuesVar1	1	Used with cbDecSimOption to specify the number of test values to use for the first selected decision variable
For Index = 3, cbDecSimOption: Integer	cbDecNumValuesVar2	2	Used with cbDecSimOption to specify the number of test values to use for the second selected decision variable
For Index = 3, cbDecSimOption: Integer	cbDecSimTrials	3	Used with cbDecSimOption to specify the maximum number of trials for each simulation
For Index = 4, cbDecForeOption: Integer	cbDecShowDefinedFore	1	Used following cbDecForeOption; the equivalent of the Show Forecasts As Defined dialog setting
For Index = 4, cbDecForeOption: Integer	cbDecShowTargetFore	2	Used following cbDecForeOption; the equivalent of the Show Only Target Forecast dialog setting
For Index = 4, cbDecForeOption: Integer	cbDecHideFore	3	Used following cbDecForeOption; the equivalent of the Hide All Forecasts dialog setting

Table 48 CB.DecisionTableND Value Parameter – Optional, Variant

Related Value	Named Constant Value	VBA Data Type	Description
For *Index2* = cbDecNumValuesVar1 or cbDecNumValuesVar2: Integer	A positive whole number	Integer	Specifies the number of test values to use for each selected decision variable
For *Index2* = cbDecSim Trials: Long Integer	A positive whole number	Long integer	Specifies the maximum number of trials for each simulation

CB.DecisionTableND Example

This example resets the simulation, defines cell B5 as the target forecast with cells C1 and C2 as the first and second decision variables. Then, it specifies the number of test values to use for each of the two decision variables (7 and 10, respectively), sets 500 as the maximum number of trials to run, and indicates that forecast charts are to be hidden while the simulation is running. The final line runs the tool.

```
CB.ResetND

'Select a target forecast:
Range("B5").Select
CB.DecisionTableND cbDecChooseTarget

'Clear the list of any previously selected decision variables
CB.DecisionTableND cbDecChooseDecVar, cbChdClearList

'Select decision variables (cells C1 and C2)
Range("C1").Select
CB.DecisionTableND cbDecChooseDecVar, cbChdAdd
Range("C2").Select
CB.DecisionTableND cbDecChooseDecVar, cbChdAdd

'Set number of values to test for DecVar1 (in cell C1)
CB.DecisionTableND cbDecSimOption, cbDecNumValuesVar1, 7

'Set number of values to test for DecVar2 (in cell C2)
CB.DecisionTableND cbDecSimOption, cbDecNumValuesVar2, 10

'Set maximum number of trials per simulation
CB.DecisionTableND cbDecSimOption, cbDecSimTrials, 500

'Hide forecast chart windows during simulation
CB.DecisionTableND cbDecForeOption, cbDecHideFore

'Run the Decision Table tool
CB.DecisionTableND cbDecRun
```

CB.DecisionTableND Related Calls

Name	Description
CB.DecisionTable	Launches the Decision Table tool

Name	Description
CB.GetDecisionTableOption	Returns current settings for the Decision Table tool

CB.DefineAltParms

This function creates or changes an alternate parameter set for continuous distributions (listed in Table 53). The function returns a value used by CB.DefineAssumND.

Note: If you use CB.DefineAltParms to define an alternate parameter set using only percentiles, you will not be able to retrieve parameter values with CB.GetAssum.

Table 49 CB.DefineAltParms Returned Data Type

Returned Value	Returned Data Type
An encoded integer that indicates the distribution and parameter set that CB.DefineAssumND should use	Integer

Note: A return value of -1 indicates an improperly defined alternate parameter set.

Table 50 CB.DefineAltParms Parameters

Parameter	VBA Data Type	Value	Description
DistType	Integer	Integer value or named constant. See Table 53.	Specifies a distribution type for the assumption. The number of parameters varies with the distribution type. See Table 53 for types and associated parameters.
Parm1, *Parm2*, *Parm3* (All are required)	Double	Double value or named constant. See Table 51, following.	Specifies the alternate parameters used to define the distribution. You must supply three parameters whether or not the distribution needs three. That is, if a distribution uses only one or two parameters, you must still supply three, using the cbParmNone parameter to round out the argument list. For more information, see Table 51.
Parm4 (Optional)	Variant	Named constant or other value	Specifies an optional fourth alternate parameter, currently required only for the beta distribution

The following constant and index values (shown in Table 51) can be used with *Parm1*, *Parm2*, *Parm3*, and *Parm4* described in Table 50.

Table 51 CB.DefineAltParms Parm1 Through Parm4 Constant and Index Values

Named Constant	Index Value	Description
cbParmNone	-1	Placeholder in the parameter list for distributions that require fewer than three parameters. This parameter must come at the end of the argument list. See the examples for how to use this parameter.

Named Constant	Index Value	Description
cbParmRegular	-2	Regular parameter for the distribution (for example, the mean or standard deviation for the normal distribution). You can't use this parameter for lognormal distributions.
cbParmLogMean	-3	Log mean
cbParmLogStDev	-4	Log standard deviation
cbParmGeoMean	-5	Geometric mean
cbParmGeoStDev	-6	Geometric standard deviation
cbParmMean	-7	Mean
cbParmStDev	-8	Sandard deviation
cbParmMinimum	-9	Mnimum value
cbParmLikeliest	-10	Lieliest value
cbParmMaximum	-11	Maximum value
cbParmLocation	-12	Location parameter. See Table 53.
cbParmScale	-13	Scale parameter. See Table 53.
cbParmMode	-15	Mode
cbParmAlpha	-16	Alpha parameter. See Table 53.
cbParmBeta	-17	Beta parameter. See Table 53.
cbParmShape	-18	Shape parameter. See Table 53.
cbParmRate	-19	Rate value. See Table 53.
cbParmMidpoint	-20	Midpoint value. See Table 53
cbParmDegFreedom	-21	Degrees of freedom value. See Table 53
cbParmPercentile	0	Percentile value. Add the desired percentile to this parameter; for example, cbParmPercentile + 80 indicates the 80th percentile.

Note: Because there are three alternate parameter sets (in addition to percentiles) for the lognormal distribution, the cbParmRegular named constant cannot be passed to CB.DefineAltParms when setting up an alternate parameter set for the lognormal distribution. You must explicitly set the parameter set with, for example, cbParmLogMean and cbParmLogStDev. cbParmMode is included to support the extreme value distribution for backward compatibility with Crystal Ball 2000.*x* (5.*x*). In later versions of Crystal Ball , that distribution is replaced with the maximum extreme and minimum extreme distributions. cbParmMode is deprecated in this version of Crystal Ball and is obsolete in a future version.

CB.DefineAltParms Example 1

This example defines the assumption in cell B5 as a normal distribution using the mean and standard deviation. Because the normal distribution requires only these two parameters, use cbParmNone to stand in as the third (unused) parameter in the argument list.

CB.DefineAssumND receives this information about the distribution and parameter set and defines the assumption's distribution with a mean of 10 and a standard deviation of 0.2.

```
Range("B5").Select
Value = CB.DefineAltParms(cbDfaNormal,cbParmMean,cbParmStDev,cbParmNone)
CB.DefineAssumND Value, 10, 0.2
```

CB.DefineAltParms Example 2

This example defines the assumption in cell B6 as a normal distribution using the 10th and 90th percentiles. The 10th percentile is 10, the 90th percentile is 12, and cbParmNone is a placeholder in the argument list.

```
Range("B6").Select
Value1 = CB.DefineAltParms(cbDfaNormal,cbParmPercentile + 10, _
   cbParmPercentile + 90, cbParmNone)
CB.DefineAssumND Value1, 10, 12
```

Note: If you use the numeric value of the index instead of the constant name for cbParmPercentile, as in Example 3, you must use the sample syntax, 0 +<percentile>, so that the function can identify the parameter by the numeric value.

CB.DefineAltParms Example 3

This example selects the assumption in cell B7 and defines a triangular distribution using the 25th percentile, the likeliest value, and the 75th percentile. CB.DefineAssumND receives this information about the distribution and parameter set and defines the assumption's distribution with a 25th percentile value of 5, a likeliest value of 10, and a 75th percentile value of 15.

Note: If you use the numeric value of the index instead of the constant name for cbParmPercentile, as in Example 3, you must use the sample syntax, 0 +*percentile*, so that the function can identify the parameter by the numeric value.

```
Range("B7").Select
Value1 = CB.DefineAltParms(cbDfaTriangular, 0 +25, -10, 0 +75)
CB.DefineAssumND Value1, 5, 10, 15
```

CB.DefineAltParms Related Calls

Name	Description
CB.DefineAssumND	Defines or changes the assumption in the selected cell
CB.GetAssum	Retrieves information for a specific assumption cell
CB.SetAssum	Changes certain attributes for the assumption in a cell

CB.DefineAssum

This subroutine opens either the Distribution Gallery to define new assumptions or the existing distribution dialogs to change existing assumptions in the selected cell.

The selected cell must be a blank cell or a simple value cell. You cannot define an assumption in a cell with a formula, a cell reference, or text.

If a simulation runs before this subroutine is used, the simulation must be reset before the assumption is defined.

Note: Before calling this subroutine, reset the simulation and select a cell.

CB.DefineAssum Example 1

This example selects cell B5, a single-value cell with no defined assumption, and then opens the Distribution Gallery so you can select a distribution for the assumption.

```
CB.ResetND
Range("B5").Select
CB.DefineAssum
```

CB.DefineAssum Example 2

This example selects cell D5, previously defined as a uniform distribution assumption, and then opens the Uniform Distribution dialog with the current parameter fields filled in. You can change the parameters or select a different distribution by clicking Gallery.

```
CB.ResetND
Range("D5").Select
CB.DefineAssum
```

CB.DefineAssum Related Calls

Name	Description
CB.CorrelateND	Defines a correlation coefficient between two selected assumptions

Name	Description
CB.DefineAssumND	Defines or modifies assumptions in selected cells without using a dialog
CB.GetAssum	Retrieves information for a specific assumption cell
CB.SetAssum	Modifies certain attributes for the assumption in a cell

CB.DefineAssumND

This subroutine defines a new assumption or changes an existing assumption in the selected cell without using a dialog.

If a simulation runs before this subroutine is used, the simulation must be reset before the assumption is defined.

Note: Before calling this subroutine, select a single cell.

Table 52 CB.DefineAssumND Parameters

Parameter	VBA Data Type	Value	Description
DistType	Integer	Integer value or named constant. See Table 53.	Specifies a distribution type for the assumption. The number of parameters varies with the distribution type. See the table on the next page for types and associated parameters.
Parm1, *Parm2*, *Parm3* (All are optional)	Variant	See Table 53.	Parameter values for the distribution type specified with the DistType parameter
LowCutOff (Optional)	Variant	Numeric	Sets the lower truncation value. The default is –Infinity. This parameter is ignored for custom distributions.
HighCutOff (Optional)	Variant	Numeric	Sets the upper truncation value. The default is +Infinity. This parameter is ignored for custom distributions.
NameOf (Optional)	Variant	Title surrounded by quotes ("")	Names the assumption. If you do not't specify a name, Crystal Ball uses the first of either: the cell's range name, text from the cell preceding (to the left of) that cell, text from the cell one row before (above) that cell, or the cell's coordinates.
Dynamic (Optional)	Variant	True or False	Specifies whether Crystal Ball calculates cell parameters once at the beginning of the simulation or with each trial
Parm4 (Optional)	Variant	See Table 53.	The minimum value for the 4-parameter beta distribution

Note: The optional Dynamic parameter is deprecated in this release. It remains in the code for compatibility with 2000.5 (5.5) versions of the Crystal Ball Developer Kit but will be obsolete in later versions because cell references are always dynamic in later versions of Crystal Ball. Beginning with version 11.1.1.0.00, the lognormal distribution has three parameters instead of two. If this call is used in a model that was created in an earlier version of Crystal Ball, the third parameter has a value of 0.

The following table lists the values for `Parm1`, `Parm2`, `Parm3`, and `Parm4` for each DistType constant. For more information about these parameter values, see the description of each distribution type in Appendix A of the current *Oracle Crystal Ball User's Guide*. The current *Oracle Crystal Ball Statistical Guide* lists defaults for each parameter value.

Table 53 CB.DefineAssumND Parameter Values and Descriptions

DistType Parameter Named Constant Value (C = continuous, D = discrete)	DistType Parameter Index Value	Parm1 Description	Parm2 Description	Parm3 Description	Parm4 Description (Optional)
cbDfaNormal (C)	0	mean value	standard deviation value	n/a	n/a
cbDfaTriangular (C)	1	minimum value	likeliest value	maximum value	n/a
cbDfaPoisson (D)	2	rate value (between 0 and 1000)	n/a	n/a	n/a
cbDfaBinomial (D)	3	probability (between 0 and 1)	trials (a whole number greater than 0 and less than 1000)	n/a	n/a
cbDfaLogNormal (C)	4	mean value	standard deviation value	location	n/a
cbDfaUniform (C)	5	minimum value	maximum value	n/a	n/a
cbDfaExponential (C)	6	rate (greater than 0)	n/a	n/a	n/a
cbDfaGeometric (D)	7	probability (between 0 and 1)	n/a	n/a	n/a
cbDfaWeibull (C)	8	location	scale (greater than 0)	shape (greater than 0.05)	n/a
cbDfaBeta (C)	9	alpha (greater than 0.3, alpha + beta must be less than 1000)	beta (greater than 0. 3, alpha + beta must be less than 1000)	maximum value	minimum value
cbDfaHypergeometric (D)	10	probability (between 0 and 1)	trials (whole number less than population)	population (whole number greater than 0 and less than 1000)	n/a

DistType Parameter Named Constant Value (C = continuous, D = discrete)	DistType Parameter Index Value	Parm1 Description	Parm2 Description	Parm3 Description	Parm4 Description (Optional)
cbDfaCustom (See "Defining Custom Distributions" on page 104) (C/D)	11	cumulative (True) or noncumulative (False). False is the default.	sequential sampling (True) or nonsequential sampling (False); see the section on SIPs in the *Oracle Crystal Ball User's Guide*	n/a	n/a
cbDfaGamma (C)	12	location	scale (greater than 0)	shape (greater than 0.05 and less than 1000)	n/a
cbDfaLogistic (C)	13	mean value	scale (greater than 0)	n/a	n/a
cbDfaPareto (C)	14	location (greater than 0)	shape (greater than 0.05)	n/a	n/a
cbDfaExtremeValue (C)	15	likeliest	scale (greater than 0)	max (True) or min (False)	n/a
cbDfaNegBinomial (D)	16	probability (between 0 and 1)	shape (whole number greater than 0 and less than 1000)	n/a	n/a
cbDfaHypergeometricSuccess (D)	17	success	trials (whole number less than population)	population (whole number greater than 0 and less than 1000)	n/a
cbDfaMinExtreme (C)	18	likeliest	scale (greater than 0)	n/a	n/a
cbDfaMaxExtreme (C)	19	likeliest	scale (greater than 0)	n/a	n/a
cbDfaStudentsT (C)	20	midpoint	scale (greater than 0)	degrees of freedom (integer between 1 and 30, inclusive)	n/a
cbDfaYesNo (D)	21	probability	n/a	n/a	n/a
cbDfaDiscreteUniform (D)	22	minimum (integer)	maximum (integer)	n/a	n/a
cbDfaBetaPert (C)	23	minimum	likeliest	maximum	n/a

Note: cbDfaHypergeometric is deprecated. It is included for Crystal Ball 2000.5 (5.5) compatibility and is replaced by cbDfaHypergeometricSuccess in later versions of Crystal Ball. cbDfaExtremeValue is deprecated. It is included for Crystal Ball 2000.5 (5.5) compatibility and is replaced by cbDfaMinExtreme and cbDfaMaxExtreme in later versions of Crystal Ball.

To define an assumption parameter with a dynamic cell reference, specify the cell reference as a string. (See the *Oracle Crystal Ball User's Guide* for more information on distribution types and parameters.)

The active cell must contain a numerical value or be blank. If it contains anything other than a numerical value (such as an expression, a cell reference, or text), CB.MacroResult returns cbErrBadAssumption.

The parameters must agree with the specified distribution. If there is a problem with a parameter, CB.MacroResult returns cbErrBadAssumption.

Note: Calling this function is not the same thing as entering an external assumption function such as CB.Normal into a worksheet cell. External assumptions are actual functions that return values using normal Microsoft Excel syntax, but you cannot correlate them with other assumptions; nor do they appear in any reports or sensitivity analyses. Calling this function, CB.DefineAssumND, is equivalent to selecting Define, then Define Assumption.

CB.DefineAssumND Example 1

This example defines a uniformly distributed assumption with a minimum value of 30, a maximum value of 40, and names it Number Of Units. This example assumes you have already selected a single cell.

```
CB.DefineAssumND cbDfaUniform,30,40,,,,"Number of Units"
```

CB.DefineAssumND Example 2

This example defines a normally distributed assumption with the following characteristics and places it in the currently selected cell:

- A mean of 15,000
- A standard deviation of 1000
- The third parameter left blank (since the normal distribution only uses two parameters)
- A lower truncation point defined by the value in cell B50
- An upper truncation point of 16,100
- A title of Expenses

```
CB.DefineAssumND cbDfaNormal,15000,1000,,"=B50",16100,"Expenses"
```

Defining Custom Distributions

To define a custom distribution, your macro code must first set the cell selection to point to both the assumption cell *and* a table containing the custom data points. You can do this using the Union object with the Range object to append the table's cell range to the assumption's cell location.

Even if set, *`LowCutOff`* and *`HighCutOff`* are ignored for custom distributions.

Parm1 can only be set to cumulative (True) if the table of data points has two, three, or four columns.

Call CB.MacroResult afterwards to determine whether the command was successful.

For more information about defining custom distributions using a data range, see the *Oracle Crystal Ball User's Guide*.

CB.DefineAssumND Custom Distribution Example

This example selects cell B3 and the range D17 to G22. CB.DefineAssumND interprets the data located in this second range of cells using the format described in the discussion of custom distributions in the *Oracle Crystal Ball User's Guide*.

```
Union(Range("B3"), Range("D17:G22")).Select
CB.DefineAssumND cbDfaCustom, False,,,,,"Sales"
```

CB.DefineAssumND Related Calls

Name	Description
CB.CorrelateND	Defines a correlation coefficient between two selected assumptions
CB.DefineAssum	Defines or changes assumptions in selected cells using a dialog
CB.GetAssum	Retrieves information for a specific assumption cell
CB.SetAssum	Changes certain attributes for the assumption in a cell

CB.DefineAssumFromForeND

This subroutine defines a new assumption from an existing forecast and places it in a specified cell without using a dialog. For easiest use, review the section on defining assumptions from forecasts in the *Oracle Crystal Ball User's Guide*.

A simulation must run before this call is used.

Table 54 CB.DefineAssumFromForeND Parameters

Parameter	VBA Data Type	Value	Description
ForeReference	Variant	A Microsoft Excel range: Range("B3")	Specifies the forecast cell to be used to define the assumption cell
AssumReference	Variant	A Microsoft Excel range: Range("B3")	Specifies the desired location for the new assumption cell

Parameter	VBA Data Type	Value	Description
ForeDistType	Integer	Integer value or named constant	Specifies whether the distribution type is determined by fitting to the forecast distribution using current fit options (set with the user interface or CB.Fit) or is a custom distribution. Values can be: • cbAsmFromFstDistTypeBestFit [1], best fit distribution parameters, or • cbAsmFromFstDistTypeCustom [2], custom distribution from forecast values
NameOf (Optional)	String	Title surrounded by quotes ("")	Optional name to use for the new assumption; can be a valid cell address (such as "=A1") to define a cell reference for the name.
UseParmRef (Optional)	Boolean	True or False	If True, creates cell references in the worksheet for distribution parameters; default is False
ParmRefFillDown (Optional)	Boolean	True or False	If True (the default), fills parameter reference cells down; if False, fills them to the right (moving from lower to higher columns)
ParmRefLabel (Optional)	Boolean	True or False	If True, includes parameter name labels adjacent to parameter value cells for all referenced parameters; default is False
TruncateCustom (Optional)	Boolean	True or False	When defining an assumption from a forecast using the Crystal Ball user interface, if the forecast trial count exceeds 5,000, users are prompted to use the first 5,000 trials or all trials. This constant is the equivalent of the prompt; if True, the first 5,000 trials are used; if False (the default), all trials are used.
SequentialCustom (Optional)	Boolean	True or False	If True, defines a sequential custom distribution retaining generated forecast value order; default is False

CB.DefineAssumFromForeND Example

The following example creates an assumption with a gamma distribution, creates a forecast with those parameters, runs a simulation, and then defines an assumption from the forecast. It then defines a custom distribution from the forecast and retains the sequence of values for sequential sampling.

```
Sub DefineAssumptionFromForecast()

    ' Define a Gamma distribution assumption and forecast to fit.
    [A1].Select
    CB.DefineAssumND cbDfaGamma, 0, 1, 2
    [B1] = "=A1"
    [B1].Select
    CB.DefineForeND "Forecast to Fit"

    ' Run 1000 trial simulation.
    CB.Simulation 1000

    ' Define an assumption from best fit distribution for forecast in cell B1.
    ' Extract the assumption parameters as cell references with
    '   parameter name labels.
    CB.DefineAssumFromForeND [B1], [D1], cbAsmFromFstDistTypeBestFit, _
       "Best Fit", True, True, True
```

```
    ' Define a custom assumption with the values from forecast in B1.
    ' Retain the sequence of generated values to sample sequentially.
    CB.DefineAssumFromForeND [B1], [E1], cbAsmFromFstDistTypeCustom, _
       "Sequential Custom Values", SequentialCustom:=True

    ' Reset simulation to view new assumptions
    CB.ResetND

End Sub
```

CB.DefineAssumFromForeND Related Calls

Name	Description
CB.DefineAssum	Defines or changes assumptions in selected cells using a dialog
CB.GetAssum	Retrieves information for a specific assumption cell
CB.SetAssum	Changes certain attributes for the assumption in a cell

CB.DefineDecVar

This subroutine opens the Define Decision Variable dialog to create a new decision variable or change an existing decision variable in the selected cell.

The selected cell must be blank or a simple value cell. You cannot define a decision variable in a cell with a formula, a cell reference, or text.

If a simulation runs before this subroutine is used, the simulation must be reset before the decision variable is defined.

Note: Before calling this subroutine, reset the simulation and select a cell.

CB.DefineDecVar Example

This example selects cell B5, a single value cell, and then opens the Define Decision Variable dialog so you can define or redefine a decision variable.

```
CB.ResetND
Range("B5").Select
CB.DefineDecVar
```

CB.DefineDecVar Related Calls

Name	Description
CB.DefineDecVarND	Defines or changes the decision variable in a selected cell without using a dialog

Name	Description
CB.GetDecVar	Retrieves information for a specific decision variable cell
CB.SetDecVar	Changes certain attributes for a decision variable

CB.DefineDecVarND

This subroutine defines or changes a decision variable in the selected cell without using a dialog.

If a simulation runs before this subroutine is used, the simulation must be reset before the decision variable is defined.

Note: Before calling this subroutine, select a cell.

Table 55 CB.DefineDecVarND Parameters

Parameter	VBA Data Type	Value	Description
Min	Variant	Any positive or negative number	Specifies the minimum (lower) bound of the decision variable. Must be an integer for the Category decision variable type; ignored for decision variable types Binary and Custom.
Max	Variant	Any positive or negative number equal to or greater than *Min*	Specifies the maximum (upper) bound of the decision variable. Must be an integer for the Category decision variable type; ignored for decision variable types Binary and Custom.
StepSize (Optional)	Variant	Number less than *Max* – *Min*	Specifies a positive step size for discrete decision variables. If *decVarType* is not used, indicates whether the decision variable type is Continuous (*StepSize* is 0 or blank) or Discrete (*StepSize* is greater than 0); ignored for decision variable types Binary, Category, and Custom.
NameOf (Optional)	Variant	Title surrounded by quotes ("").	Specifies how Crystal Ball identifies the decision variable. Names the decision variable. If you do not specify a name, Crystal Ball uses the first of either: the cell's range name, text from the cell in the column immediately preceding it (to the left), text from the cell in the row immediately preceding it (above), or the cell's coordinates.
decVarType (Optional)	Integer	See Table 56.	Sets the decision variable type to Continuous, Discrete, Binary, Category, or Custom (described in Table 56).
customData (Optional)	String	A list of at least two values separated by commas (,) or a reference to a range of cells containing these values; decimal values can be used. The range must be one-dimensional (adjoining cells in one row *or* one column).	Supplies a list of values for use with the Custom decision variable type (cbDecVarTypeCustom).

Note: The decVarType and customData parameters are added for Crystal Ball 11.*x*.

Table 56 Named Constant and Index Values for the decVarType Parameter — Optional, Integer

Named Constant Value	Index Value	Description
cbDecVarTypeContinuous	0	Sets the decision variable type to Continuous; it can assume any value between the lower and upper bounds. Continuous is the default type if *decVarType* is not used and *StepSize* is blank or 0.
cbDecVarTypeDiscrete	1	Sets the decision variable type to Discrete; it can assume values at specific intervals, based on step size, between the lower and upper bounds. If *decVarType* is not used and *StepSize* is neither blank nor 0, the decision variable type is set to Discrete.
cbDecVarTypeBinary	2	Sets the decision variable type to Binary; it can assume the value 0 (represents No) or 1 (represents Yes). Also known as the "Yes-No" type.
cbDecVarTypeCategory	3	Sets the decision variable type to Category; it can assume any integer between the upper and lower bounds (inclusive) without regard to order. The bounds are also integers. The decision variable values are nominal and represent choices or categories instead of arithmetic values. For more information, see CB.DefineDecVarND Example 4.
cbDecVarTypeCustom	4	Sets the decision variable type to Custom; it can assume any value from a list of two or more specific values supplied with the *customData* parameter.

CB.DefineDecVarND Example 1

This example defines a new decision variable in cell C9 with a lower bound of 400, an upper bound of 600, a step size of 10 (also making the decision variable Discrete), and a title of Rent Per Unit.

```
Range("C9").Select
CB.DefineDecVarND 400, 600, 10, "Rent Per Unit"
```

This definition is the same except the decVarType parameter is used:

```
Range("C9").Select
CB.DefineDecVarND 400, 600, 10, "Rent Per Unit", cbDecVarTypeDiscrete
```

CB.DefineDecVarND Example 2

This example defines a new decision variable in cell D12 with a lower bound of 500, an upper bound of 800, no step size (also defining the decision variable as Continuous), and a title of Net Income.

```
Range("D12").Select
CB.DefineDecVarND 500, 800,, "Net Income"
```

This definition is the same except the decVarType parameter is used:

```
Range("D12").Select
CB.DefineDecVarND 500, 800,, "Net Income", cbDecVarTypeContinuous
```

CB.DefineDecVarND Example 3

This example defines a new decision variable in cell F10 with a lower bound of 0, an upper bound of 1, no step size, a title of Net Income, and a decision variable type of Binary. The values of Min and Max are ignored for Binary decision variables, but they must be entered since they are required parameters for CB.DefineDecVarND. If a step size is entered for this decision variable type, it is ignored.

```
Range("F10").Select
CB.DefineDecVarND 0, 1,, "Net Income", cbDecVarTypeBinary
```

CB.DefineDecVarND Example 4

Crystal Ball example model Groundwater Cleanup.xls contains two decision variables. The decision variable in cell D13 is named Remediation Method. It has a lower bound of 1.00, an upper bound of 3.00, and is defined as a Discrete decision variable with a step size of 1.00. Notice that this definition yields three values, expressed as integers 1, 2, and 3. These are not numeric values, but instead they represent three different remediation methods for groundwater cleanup.

This example model was included with Crystal Ball before new decision variable types were added. Starting with Crystal Ball 11.1.1.0.00, this decision variable can be defined as the Category type, shown in the following code:

```
Range("D13").Select
CB.DefineDecVarND 1, 3, 1, "Remediation Method", cbDecVarTypeCategory
```

This code defines a new decision variable in cell D13 that is identical to the one in Groundwater Cleanup.xls except for the decision variable type. It has a lower bound of 1, an upper bound of 3, a step size of 1, a title of Remediation Method, and a decision variable type of Category. Although a step size is entered for this decision variable type, it is ignored.

CB.DefineDecVarND Example 5

This example defines a new decision variable in cell G8 with a lower bound of 10, an upper bound of 40, no step size, a title of Net Income, and a decision variable type of Custom. The values of *Min* and *Max* are ignored for Custom decision variables, but they must be entered since they are required parameters for CB.DefineDecVarND. If a step size is entered for this decision variable type, it is ignored.

```
Range("G8").Select
CB.DefineDecVarND 10,40,,"Net Income",cbDecVarTypeCustom, "10,20,30,40"
```

Note that the data string (the *customData* parameter) can also be a valid cell range. If the data string in the previous code is stored in range A1:D1, the code becomes:

```
Range("G8").Select
CB.DefineDecVarND 10,40,,"Net Income",cbDecVarTypeCustom,"A1:D1"
```

CB.DefineDecVarND Related Calls

Name	Description
CB.DefineDecVar	Defines or changes the decision variable in a selected cell using a dialog
CB.GetDecVar	Retrieves information for a specific decision variable cell
CB.SetDecVar	Changes certain attributes for a decision variable

CB.DefineFore

This subroutine opens the Define Forecast dialog to create a new forecast or change an existing forecast in the selected cell.

The selected cell can be a simple value or formula cell. You cannot define a forecast in a cell with an assumption.

Note: Before calling this subroutine, select a cell or cell range.

CB.DefineFore Example

This example opens the Define Forecast dialog so you can define or redefine a forecast.

```
Range("B5").Select
CB.DefineFore
```

CB.DefineFore Related Calls

Name	Description
CB.GetFore	Retrieves information for a specific forecast cell
CB.SetFore	Changes forecast attributes without redefining the forecast

CB.DefineForeND

This subroutine defines or changes a forecast in the selected cell without using a dialog.

The selected cell must contain a simple value, an expression, or a cell reference; otherwise, CB.MacroResult returns cbErrNoForecastInCell.

Note: Before calling this subroutine, select a cell.

Table 57 CB.DefineForeND Parameters

Parameter	VBA Data Type	Value	Description
FName (Optional)	Variant	Title surrounded by quotes ("")	Specifies how Crystal Ball identifies the forecast. If you do not specify a name, Crystal Ball uses the first of either: the cell's range name, text from the cell in the column immediately preceding it (to the left), text from the cell in the row immediately preceding it (above), or the cell's coordinates
Units (Optional)	Variant	Words surrounded by quotes ("")	Specifies the units label of the forecast chart
DisplayAuto (Optional)	Variant	True or False	True sets the forecast window to open automatically according to the *DisplayWhileRunning* setting. False requires you to open the forecast chart manually through the Analyze, then Forecast Charts command. The default is True.
DisplayWhileRunning (Optional)	Variant	True or False	Used only when *DisplayAuto* is set to True. True for this parameter opens the forecast window at the start of a simulation. False opens the forecast window at the end of a simulation. The default is True.
Obsolete_WindowSize (Optional)	n/a	n/a	This parameter, formerly *WindowSize*, is obsolete and is no longer available for use. However, if you use one of the following optional parameters, you need to leave a space for it set off by a comma.
LSL (Optional)	Variant	A numeric value (double), a string representing a cell reference (such as "=B1"), or cbParmUndefined	Enters a lower specification limit for the forecast
USL (Optional)	Variant	A numeric value (double), a string representing a cell reference (such as "=B1"), or cbParmUndefined	Enters an upper specification limit for the forecast
Target (Optional)	Variant	A numeric value (double), a string representing a cell reference (such as "=B1"), or cbParmUndefined	Enters a target value for the forecast

CB.DefineForeND Example

This example defines a new forecast in cell C9 with the name of Profit and no unit label. The forecast chart window is set to appear automatically when the simulation starts.

```
Range("C9").Select
CB.DefineForeND "Profit", "", True, True
```

CB.DefineForeND Related Calls

Name	Description
CB.DefineFore	Defines forecasts in selected cells using a dialog
CB.GetFore	Retrieves information for a specific forecast cell
CB.SetFore	Changes forecast attributes without redefining the forecast

CB.DeleteChart

This call deletes a chart of the specified type and chart ID. The chart ID is optional. If the chart ID is not specified, the currently selected chart is deleted. If the specified chart is not found, a chooser dialog appears. No error is returned.

Table 58 CB.DeleteChart Parameters and Constants

Parameter	VBA Data Type	Value	Description
SimChartType	Integer	cbChtOverlay = 1, cbChtScatter = 2, cbChtSensitiv = 3, cbChtTrend = 4	Specifies the type of chart you are deleting: overlay, scatter, sensitivity, or trend. For named constant and index values, see the Value column.
ChartID (Optional)	String	"*Chart ID*"	The chart ID returned by CB.EnumChart (Table 60) entered directly or using CB.EnumChart

CB.DeleteChart Example

This example deletes the overlay chart, trend chart, and sensitivity chart specified with CB.EnumChart.

```
CB.CheckData
CB.DeleteChart cbChtOverlay, CB.EnumChart(cbChtOverlay)
CB.DeleteChart cbChtTrend, CB.EnumChart(cbChtTrend)
CB.DeleteChart cbChtSensitiv, CB.EnumChart(cbChtSensitiv)
```

CB.DeleteChart Related Calls

Name	Description
CB.CreateChart	Creates a new chart of the specified type with the optional name, if specified
CB.EnumChart	Returns the chart ID of the next chart of the specified type
CB.SelectChart	Selects the chart with the specified chart ID

CB.EnumAssum

This function enumerates assumption cells for all open workbooks. This command is useful for determining what assumptions have been defined for the open workbooks. You can then call CB.GetAssum for each assumption to retrieve specific information—for example, distribution type and parameters.

Table 59 CB.EnumAssum Returned Data Type

Returned Value	Returned Data Type
The absolute cell reference of the next assumption, in the format: '[workbook]sheet'!$A$1	Variant

Note: For best results, call CB.CheckData once at the beginning of an enumeration to reset the sequence of assumptions. You should also call CB.CheckData each time you add or delete data. Use CB.CheckData or CB.CheckDataND with CB.MacroResult and CB.MacroResultDetail to check for errors. For more information, see the descriptions of those calls.

If the return string is empty, all assumptions have been enumerated. You can use the results of this subroutine to create a Range object.

You can only call this function from a subroutine or another function. You cannot call this function from a worksheet.

CB.EnumAssum Example

This example initializes all internal variables, then steps through all the assumptions, creating range objects from the returned string and putting their absolute cell references in column B of a worksheet.

```
CB.CheckData
Dim S As String
Dim R As Range
Dim t As Integer
S=CB.EnumAssum
While S<>""
   Set R=Range(S)
   Range("B2").Offset(t).Value = s
   t = t + 1
   S=CB.EnumAssum
Wend
```

CB.EnumAssum Related Calls

Name	Description
CB.EnumDecVar	Enumerates decision variable cells for all open workbooks

Name	Description
CB.EnumFore	Enumerates forecast cells for all open workbooks
CB.GetAssum	Retrieves information for a specific assumption cell
CB.GetAssumPercent	Returns percentile for an assumption cell

CB.EnumChart

This function returns the unique name of the next chart of the specified type. The string returned is the full path of the chart. You can then use other window and chart handling calls to manage the enumerated charts.

Table 60 CB.EnumChart Returned Data Type

Returned Value	Returned Data Type
The path and name of the next chart of that type, ChartID, in the format: "|Current Results|Sensitivity Charts|Charts.xls|Sensitivity Chart 1"	String

Note: For best results, call CB.CheckData once at the beginning of an enumeration to reset the sequence of charts. You should also call CB.CheckData each time you add or delete data. Use CB.CheckData or CB.CheckDataND with CB.MacroResultDetail to check for errors. For more information, see the descriptions of those calls.

This call operates only on current results and not restored results.

If the returned string is empty, either all charts of that type have been enumerated or there were no charts of that type to enumerate.

You can only call this function from a subroutine or another function. You cannot call this function from a worksheet.

CB.EnumChart has only one parameter, *SimChartType*, which specfies the type of chart to enumerate.

Table 61 CB.EnumChart SimChartType Parameter Values — Required, Integer

Named Constant	Index	Description
cbChtOverlay	1	Enumerates overlay charts
cbChtScatter	2	Enumerates scatter charts
cbChtSensitiv	3	Enumerates sensitivity charts
cbChtTrend	4	Enumerates trend charts

CB.EnumChart Example

This example initializes all internal variables, then steps through all the overlay charts, returns their paths and names, and puts this information in column C of a worksheet.

```
CB.Simulation 1000
CB.CheckData
Dim ChartID As String
Dim t As Integer
ChartID = CB.EnumChart(cbChtOverlay)
t = 0
While ChartID <> ""
   Range("C2").Offset(t).Value = ChartID
   t = t + 1
   ChartID = CB.EnumChart(cbChtOverlay)
Wend
```

CB.EnumChart Related Calls

Name	Description
CB.CheckData	Checks decision variables for validity, initializes variables
CB.CreateChart	Creates a new chart of the specified type with the optional name, if specified
CB.SelectChart	Selects the chart with the specified chart ID

CB.EnumCorrelation

This function lists correlated assumption cells and their correlation values for all open workbooks. This command is useful for determining what correlated assumptions have been defined for the open workbooks. You could then call CB.GetAssum for each assumption to retrieve specific information—for example, distribution type and parameters.

Table 62 CB.EnumCorrelation Returned Data Type

Returned Value	Returned Data Type
The current count of correlations left in the enumeration (including the correlation returned by this call)	Variant

Note: For best results, call CB.CheckData once at the beginning of an enumeration to reset the sequence of correlated assumptions. You should also call CB.CheckData each time you add or delete data. Use CB.CheckData or CB.CheckDataND with CB.MacroResultDetail to check for errors. For more information, see the descriptions of those calls.

If the returned string is 0, all correlated assumptions have been enumerated.

You can only call this function from a subroutine or another function. You cannot call this function from a worksheet.

Table 63 CB.EnumCorrelation Parameters

Parameter	VBA Data Type	Description
Assum1	String	Represents the first assumption in a pair of correlated assumptions, by reference, returned as the absolute cell reference of the next correlated assumption, in the format: '[workbook]sheet'!$A$1
Assum2	String	Represents the second assumption in a pair of correlated assumptions, by reference, returned as the absolute cell reference of the next correlated assumption, in the format: '[workbook]sheet'!$A$1
Value	Variant	Represents the correlation value associated with *Assum1* and *Assum2*, by reference

Notice that all parameters are "by reference." The values passed into CB.EnumCorrelation are ignored. The next set of enumeration values is assigned to the parameters when the function returns.

CB.EnumCorrelation Example 1

This example uses asm1Address to represent *Assum1*, uses asm2Address to represent *Assum2*, uses corrValue to represent *Value*, and then uses a loop to stop execution when all correlated assumptions have been located. As it runs, it displays a message box with the first and second assumption in each pair of correlated assumptions along with the correlation value.

```
Sub EnumCorr()
   Dim asm1Address As String
   Dim asm2Address As String
   Dim corrValue As Variant
   Dim count As Variant

   CB.CheckData
   ' When asm1Address, asm2Address, and corrValue are passed ByRef,
   ' the value passed into the function is ignored and the next set of
   'values in the enumeration is set in these parameters when the call    'returns.

   count = CB.EnumCorrelation(asm1Address, asm2Address, corrValue)

   While (count > 0)
      MsgBox "Assumption in cell " & asm1Address & _
         " is correlated to assumption in cell " & asm2Address & _
         " by " & corrValue
      count = CB.EnumCorrelation(asm1Address, asm2Address, corrValue)
   Wend
End Sub
```

CB.EnumCorrelation Example 2

This example uses asm1Address to represent *Assum1*, uses asm2Address to represent *Assum2*, uses corrValue to represent *Value*, and then uses a loop to stop execution when all correlated assumptions have been located.

As it runs, it produces a list of the first and second asssumption in each pair of correlated assumptions, followed by the correlation value for that pair.

```
Sub EnumCorr2()
   CB.CheckData

   Dim asm1Address As String
   Dim asm2Address As String
   Dim corrValue As Variant
   Dim count As Variant
   Dim initialCount As Variant

   initialCount = CB.EnumCorrelation(asm1Address, asm2Address, corrValue)
   count = initialCount
   While (count > 0)
      Cells(initialCount - count + 1, 5) = asm1Address
      Cells(initialCount - count + 1, 6) = asm2Address
      Cells(initialCount - count + 1, 7) = corrValue
      count = CB.EnumCorrelation(asm1Address, asm2Address, corrValue)
   Wend
End Sub
```

CB.EnumCorrelation Related Calls

Name	Description
CB.EnumAssum	Enumerates assumption cells for all open workbooks
CB.GetAssum	Retrieves information for a specific assumption cell

CB.EnumDecVar

This function enumerates decision variable cells for all open workbooks. This command is useful for determining what decision variables have been defined for the open worksheets. You can then call CB.GetDecVar for each decision variable to retrieve specific information—for example, range minimum or maximum.

Table 64 CB.EnumDecVar Returned Data Type

Returned Value	Returned Data Type
The absolute cell reference of the next decision variable, in the format:'[workbook]sheet'!$A$1	Variant

Note: For best results, call CB.CheckData once at the beginning of an enumeration to reset the sequence of decision variables. You should also call CB.CheckData each time you add or delete data. Use CB.CheckData or CB.CheckDataND with CB.MacroResultDetail to check for errors. For more information, see the descriptions of those calls.

If the return string is empty, all decision variables have been enumerated. You can use the results of this subroutine to create a Range object.

You can only call this function from a subroutine or another function. You cannot call this function from a worksheet.

CB.EnumDecVar Example

This example initializes all internal variables, then steps through all the decision variables, creates range objects from the returned string and puts their absolute cell references in column B of a worksheet.

```
CB.CheckData
Dim S As String
Dim R As Range
Dim t As Integer
S=CB.EnumDecvar
While S<>""
   Set R=Range(S)
   Range("B2").Offset(t).Value = s
   t = t + 1
   S=CB.EnumDecVar
Wend
```

CB.EnumDecVar Related Calls

Name	Description
CB.CheckData	Checks decision variables for validity, initializes variables
CB.EnumAssum	Enumerates assumption cells for all open worksheets
CB.EnumFore	Enumerates forecast cells for all open workbooks
CB.GetDecVar	Retrieves information for a specific decision variable cell

CB.EnumFore

This function enumerates forecast cells for all open workbooks. This command is useful for determining what forecasts have been defined for the open worksheets. You can then call CB.GetFore for each forecast to retrieve specific information—for example, the forecast name and units.

Table 65 CB.EnumFore Returned Data Type

Returned Value	Returned Data Type
The absolute cell reference of the next forecast, in the format: `'[workbook]sheet'!$A$1`	Variant

Note: Call CB.CheckData once at the beginning of an enumeration to reset the sequence of forecasts. Call CB.CheckData each time you add or delete data. Use CB.CheckData or CB.CheckDataND with CB.MacroResultDetail to check for errors. For more information, see the descriptions of those calls.

If the return string is empty, all forecasts have been enumerated. You can use the results of this function to create a Range object.

You can only call this function from a subroutine or another function. You cannot call this function from a worksheet.

CB.EnumFore Example

This example runs a simulation, initializes all internal variables, and then steps through all the forecasts, creates range objects from the returned string, and opens the forecast window for each one. Note that the range objects are created with absolute cell references.

```
CB.CheckData
Dim S As String
Dim R As Range
CB.Simulation 1500
s = CB.EnumFore
While s <> ""
   Set r = Range(s)
   r.Select
   CB.OpenFore
   s = CB.EnumFore
Wend
```

CB.EnumFore Related Calls

Name	Description
CB.CheckData	Checks assumptions for validity, initializes variables
CB.EnumAssum	Enumerates assumption cells for all open workbooks
CB.EnumDecVar	Enumerates decision variable cells for all open workbooks
CB.GetFore	Retrieves information for a specific forecast cell
CB.GetForePercent	Returns a percentile for a forecast cell
CB.GetForeStat	Returns a statistic for a forecast cell

CB.ExtractData

This function extracts simulation data. It opens the Extract Data dialog so you can specify the data to extract and then extract it.

Only call this function after you run a simulation, optimization, time-series forecast, or restore saved results. Use CB.CheckData before using this call.

CB.ExtractData Example

This example opens the Extract Data dialog.

```
CB.Simulation 1000
CB.CheckData
CB.ExtractData
```

CB.ExtractData Related Calls

Name	Description
CB.CreateRpt	Creates a report from the current simulation and restored results, optimization, or time-series forecast using a dialog
CB.CreateRptND	Creates a report from the current simulation and restored results, optimization, or time-series forecast without a dialog
CB.ExtractDataND	Extracts data from the current simulation and restored results, optimization, or time-series forecast without a dialog

CB.ExtractDataND

This function extracts Crystal Ball data from the the current simulation. For any preferences not explicitly set, the default is the current setting in the Extract Data dialog. Use CB.CheckData before using this call. You can extract more than one type of data. For details, see CB.ExtractDataND Example 2.

Note: Before calling this function, run a simulation, OptQuest optimization, or Predictor time-series forecast.

Table 66 CB.ExtractDataND Parameters

Parameter	VBA Data Type	Description
Index	Integer	Specifies which extract preference you are setting. See Table 67, following.
Value (Optional)	Variant	Specifies settings for the preference indicated by the Index. See Table 67 for more information.
Value2 (Optional)	Variant	Specifies whether to extract the specified data type. See CB.ExtractDataND Example 2.

The following are named constant and index values for the *Index* parameter.

Table 67 CB.ExtractDataND Index Parameter Values (Required, Integer) with Related Value Parameter Values (Optional, Variant)

Index Parameter Named Constant Value	Index Parameter Index Value	Value Parameter Values to Use	Description
cbExtOK	1	n/a	Extracts the data. Use this parameter after setting other extract options.

Index Parameter Named Constant Value	Index Parameter Index Value	Value Parameter Values to Use	Description
cbExtDataType	2	See Table 68	Specifies the type of data to extract. Used with these *Value* parameter values, described in Table 68: 1 = cbDatForeValues (CB 2000.*x* or 5.*x* only), 1 = cbDatValues (CB 7.*x* and 11.*x* or later), 2 = cbDatStatistics, 3 = cbDatPercentiles, 4 = cbDatFrequencies, 6 = cbDatSensitiv, 7 = cbDatCapMetrics.
cbExtChooseFore	3	See Table 69	Specifies which forecasts to include. Used with these *Value* parameter values, described in Table 69: 1 = cbChfAll, 2 = cbChfOpen, 3 = cbChfChosen, 4 = cbChfAdd, 5 = cbChfClearList.
cbExtPercentiles	4	See Table 70	Specifies which percentiles to extract. Used with these *Value* parameter values, described in Table 70: 1 = cbPctQuartiles, 2 = cbPctQuintiles, 3 = cbPctDeciles, 4 = cbPctIcosatiles, 5 = cbPctSet1, 6 = cbPctSet2, 7 = cbPctSet3, 8 = cbPctSet4.
cbExtExistingSheet	5	True or False	True extracts data to a new worksheet called Data in the active workbook. False extracts data to a new worksheet called Data in a new workbook.
cbExtChooseAsm	6	See Table 71	Specifies which assumptions to include. Used with these *Value* parameter values, described in Table 71: 1 = cbChaAll, 2 = cbChaChosen, 3 = cbChaAdd, 4 = cbChaClearList, 5 = cbChaOpen.
cbExtChartBins	7	A value between 10 and 1000	Indicates how many bins (value groups or intervals) to include. **Note:** This value is independent of the Chart Preferences density setting.
cbExtIncludeCellLoc	8	True or False	True includes cell location information. False does not include cell location information.
cbExtIncludeLabels	9	True or False	True includes labels. False does not include labels.
cbExtAutoFormat	10	True or False	True applies formatting to the cells as appropriate. False does not apply formatting.
cbExtSheetName	11	A string value representing a sheet name	The name of the sheet where extracted data is placed. The sheet name must follow appropriate sheet naming restrictions.
cbExtStartCell	12	A string value representing a starting cell location	Specifies the first (upper-left) cell of the extracted data.

Index Parameter Named Constant Value	Index Parameter Index Value	Value Parameter Values to Use	Description
cbExtStartSheet	13	See Table 72	Indicates where to extract the data, described in Table 72: 1 = cbExtNewWorkbook, 2 = cbExtCurrentWorkbook_ NewSheet, 3 = cbExtCurrentWorkbook_CurrentSheet. **Note:** Calling ExtractDataND with this value will overwrite any settings through cbExtExistingSheet.
cbExtDataOptimizerType	14	See Table 73	Indicates which OptQuest data to extract following an optimization run. If not explicitly set, the current settings from the Extract Data dialog are used.
cbExtChooseDecVar	15	See Table 74	Indicates which decision variable data to extract following an optimization run. If not explicitly set, the current settings from the Extract Data dialog are used.
cbExtDataPredictorType	16	Constant name or number as described in the next table cell.	Specifies which type of Predictor data to extract: ● 1 = cbDatPredictorResults, extracts results data ● 2 = cbDatPredictorMethods, extracts methods data **Note:** The constant is followed by a boolean (True or False) to indicate whether the constant is active or not.
cbExtPredictorResultsSheetName	17	A string value representing a sheet name	The name of the sheet where extracted Predictor results data is placed. The sheet name must follow appropriate sheet naming restrictions.
cbExtPredictorMethodsSheetName	18	A string value representing a sheet name	The name of the sheet where extracted Predictor methods data is placed. The sheet name must follow appropriate sheet naming restrictions.
cbExtPredictorResultsHistorical	19	True or False	True includes historical data in extracted results. False does not include historical data.
cbExtPredictorResultsFitted	20	True or False	True includes fitted data in extracted results. False does not include fitted data.
cbExtPredictorResultsForecast	21	True or False	True includes forecasted data in extracted results. False does not include forecasteddata.
cbExtPredictorResultsConfidence	22	True or False	True includes confidence interval data in extracted results. False does not include confidence interval data.
cbExtPredictorResultsResiduals	23	True or False	True includes residuals data in extracted results. False does not include residuals data. For information on residuals data, see the *Oracle Crystal Ball Predictor User's Guide*.
cbExtPredictorMethodsErrors	24	True or False	True includes methods error data in extracted results. False does not include methods error data.
cbExtPredictorMethodsStatistics	25	True or False	True includes methods statistics data in extracted results. False does not include methods statistics data.
cbExtPredictorMethodsParameters	26	True or False	True includes methods parameters data in extracted results. False does not include methods parameters data.

Index Parameter Named Constant Value	Index Parameter Index Value	Value Parameter Values to Use	Description
cbExtPredictorMethodsRanking	27	True or False	True includes methods ranking data in extracted results. False does not include methods ranking data.
cbExtChartBinsUseEntireRange	28	True or False	True uses the entire chart range, including extreme values that are not displayed. False uses only the displayed chart range.

The following values are used with *Index* = cbExtDataType to specify the type of Crystal Ball data to extract.

Table 68 Value Parameter Values that Support Index Constant cbExtDataType

Named Constant Value	Index Value	Description
cbDatForeValues	1	Extracts forecast and assumption values **Note:** The constant cbDatForeValues is included for compatibility with Crystal Ball 2000.5 (5.5) and is considered deprecated in later versions of Crystal Ball; for future compatibility, use CBDatValues instead.
cbDatValues	1	Extracts forecast and assumption values; same as cbDatForeValues but name reflects capabilities more accurately
cbDatStatistics	2	Extracts statistics
cbDatPercentiles	3	Extracts percentiles
cbDatFrequencies	4	Extracts the number of data points in each chart bin
cbDatSensitiv	6	Extracts sensitivity data
cbDatCapMetrics	7	Extracts capability metrics data. True extracts the metrics and False does not extract the metrics.
cbPctSet3	7	Extraction of the 5% and 95% percentiles

Note: The constant cbDatCumulative [5] is not included in this release.

The following values are used with *Index* = cbExtChooseFore to specify forecast data for extraction.

Table 69 Value Parameter Values that Support Index Constant cbExtChooseFore

Named Constant Value	Index Value	Description
cbChfAll	1	Includes all forecasts
cbChfOpen	2	Includes open forecasts
cbChfChosen	3	Includes chosen forecasts. Use after cbChfAdd.

Named Constant Value	Index Value	Description
cbChfAdd	4	Adds the forecast in the selected cell to the list of chosen forecasts. Use cbChfChosen after this option. **Note:** An error is returned if the user attempts to add (cbChfAdd) a forecast which contains no underlying data. During extraction, only those forecasts with underlying data are actually extracted. The others are ignored.
cbChfClearList	5	Clears the list of chosen forecasts

The following values are used with *Index* = cbExtPercentiles to specify percentile data for extraction.

Table 70 Value Parameter Values that Support Index Constant cbExtPercentiles

Named Constant Value	Index Value	Description
cbPctQuartiles	1	Extracts quartiles (25%)
cbPctQuintiles	2	Extracts quintiles (20%)
cbPctDeciles	3	Extracts deciles (10%)
cbPctIcosatiles	4	Extracts icosatiles (5%)
cbPctSet1	5	Extracts 2.5, 5, 50, 95, 97.5%-iles
cbPctSet2	6	Extracts 10, 25, 50, 75, 90%-iles
cbPctSet3	7	Extracts the 5% and 95% percentiles
cbPctSet4	8	Extracts the 10% and 90% percentiles

The following values are used with *Index* = cbExtChooseAsm to specify assumption data for extraction.

Table 71 Value Parameter Values that Support Index Constant cbExtChooseAsm

Named Constant Value	Index Value	Description
cbChaAll	1	Includes all assumptions
cbChaChosen	2	Includes chosen assumptions. Run this option after adding assumptions.
cbChaAdd	3	Adds the assumption in the selected cell to the chosen assumptions. Use cbChaChosen after this option.
cbChaClearList	4	Clears the list of chosen assumptions
cbChaOpen	5	Includes all open assumptions

The following values are used with *Index* = cbExtStartSheet to specify whether data should be extracted to a new or existing workbook and worksheet.

Note: Calling ExtractDataND with cbExtStartSheet will overwrite any settings made with cbExtExistingSheet.

Table 72 Value Parameter Values that Support Index Constant cbExtStartSheet

Named Constant Value	Index Value	Description
cbExtNewWorkbook	1	Extracts the results to a new workbook, using the cbExtSheetName value as the name of the sheet
cbExtCurrentWorkbook_NewSheet	2	Extracts the results to the current workbook using the cbExtSheetName value as the name of the sheet
cbExtCurrentWorkbook_ CurrentSheet	3	Extracts the results to the current workbook's current sheet

The following values are used with Index = cbExtDataOptimizerType to specify which optimizer data should be extracted to the specified location. A third boolean value, True or False, is used to activate or deactivate the preceding entries.

Table 73 Value Parameter Values for Index Constant cbExtDataOptimizerType

Named Constant Value	Index Value	Description
cbDatOptimizerSolutions	1	Extracts OptQuest solution information
cbDatOptimizerStatistics	2	Extracts OptQuest statistical information

The following values are used with Index = cbExtChooseDecVar to specify decision variable data for extraction. If not specified, the current setting from the Extract Data dialog is used.

Table 74 Value Parameter Values that Support Index Constant cbExtChooseDecVar

Named Constant Value	Index Value	Description
cbChooseDecVarAll	1	Includes all decision variables
cbChooseDecVarChosen	2	Includes chosen forecasts. Use after cbChooseDecVarAdd.
cbChooseDecVarAdd	3	Adds the forecast in the selected cell to the list of chosen forecasts. Use cbChooseDecVarChosen after this option.
cbChooseDecVarClearList	4	Clears the list of chosen forecasts

Run a simulation before you call CB.ExtractDataND. Extraction of data from restored results is not supported.

See the *Oracle Crystal Ball User's Guide* for more information on extracting data.

CB.ExtractDataND Example 1

This example runs a simulation and then sets preferences to extract the decile percentile data for all the forecasts. It then extracts the data to a new worksheet in the active workbook.

```
CB.Simulation 1000
CB.CheckData
CB.ExtractDataND cbExtChooseFore, cbChfAll
CB.ExtractDataND cbExtDataType, cbDatPercentiles
CB.ExtractDataND cbExtPercentiles, cbPctDeciles
CB.ExtractDataND cbExtExistingSheet, True
CB.ExtractDataND cbExtOK
```

CB.ExtractDataND Example 2

You can now use CB.ExtractDataND to extract more than one type of data. To do this, use cbExtDataType followed by the type of data, and then follow that by a boolean Value2 parameter. For example, the following lines of code extract both statistics and values:

```
CB.CheckData
CB.ExtractDataND cbExtDataType, cbDatStatistics, True
CB.ExtractDataND cbExtDataType, cbDatValues, True
CB.ExtractDataND cbExtOK
```

CB.ExtractDataND Related Calls

Name	Description
CB.CreateRpt	Creates a Crystal Ball report from the creates a report from the current simulation and restored results, optimization, or time-series forecast using a dialog
CB.CreateRptND	Creates a Crystal Ball report from the creates a report from the current simulation and restored results, optimization, or time-series forecast without a dialog
CB.ExtractData	Extracts data from the current simulation and restored results, optimization, or time-series forecast using a dialog

CB.Fit

This function fits a specific probability distribution to a set of data (range of cells) and returns information about the fit.

Table 75 CB.Fit Returned Data Type

Returned Value	Returned Data Type
The goodness-of-fit statistic for the fit of the data to the selected distribution	Variant

You select the distribution to fit to your data (among the possible distributions listed in Table 76 on page 128) and the statistic to return.

CB.Fit must be used with CB.SetFitRange ("CB.SetFitRange" on page 211), as shown in the examples for this call.

Note: You cannot fit to distributions that are not listed.

Table 76 CB.Fit Parameters

Parameter	VBA Data Type	Value	Description
Dist	Integer	See Table 77	Used to specify the distribution to use for fitting. Values are listed in Table 77.
Criterion	Integer	See Table 78	Used to specify which fitting statistics to return. Values appear in Table 78.
pValue (Optional)	Boolean	True or False	If False, returns the statistic; if True, returns the p-value. The default is False.
DistBestFit	Integer	See Table 77	If only one distribution is passed for *Dist*, *DistBestFit* returns the index of that distribution. Otherwise it returns the index of the best fitted distribution.
CriterionBestFit	Integer	See Table 78	If a specific ranking method is passed for *Criterion*, *CriterionBestFit* returns the index of this ranking method. Otherwise, it returns the index of the selected ranking method.

The following table shows Dist parameter values, used to select a distribution for fitting. You can only fit to distributions listed in this table.

Table 77 CB.Fit Dist Parameter Values — Required, Integer

Named Constant Value	Index Value	Description
cbDfaNormal	0	Fits to a normal distribution
cbDfaTriangular	1	Fits to a triangular distribution
cbDfaPoisson	2	Fits to a Poisson distribution
cbDfaBinomial	3	Fits to a binomial distribution
cbDfaLognormal	4	Fits to a lognormal distribution
cbDfaUniform	5	Fits to a uniform distribution
cbDfaExponential	6	Fits to an exponential distribution
cbDfaGeometric	7	Fits to a geometric distribution
cbDfaWeibull	8	Fits to a Weibull distribution
cbDfaBeta	9	Fits to a beta distribution
cbDfaGamma	12	Fits to a gamma distribution
cbDfaLogistic	13	Fits to a logistic distribution
cbDfaPareto	14	Fits to a Pareto distribution
cbDfaExtremeValue	15	Fits to an extreme value distribution
cbDfaNegBinomial	16	Fits to a negative binomial distribution
cbDfaHypergeometric Success	17	Fits to a hypergeometric distribution with Success parameter
cbDfaMinExtreme	18	Fits to a minimum extreme value distribution

Named Constant Value	Index Value	Description
cbDfaMaxExtreme	19	Fits to a maximum extreme value distribution
cbDfaStudentsT	20	Fits to a Student's *t* distribution
cbDfaDiscreteUniform	22	Fits to a discrete uniform distribution
cbDfaBetaPert	23	Fits to a betaPERT distribution
cbDfaAutoSelectDist	50	Fits to whatever Crystal Ball chooses in the Distribution Gallery as the best type of distribution for fitting.
cbDfaAllContinuous	51	Fits the data to all continuous distributions in the Distribution Gallery (those distributions in which every value in the distribution's range is possible).
cbDfaAllDiscrete	52	Fits the data to all discrete (non-continuous) distributions in the Distribution Gallery (except Yes-No)
cbDfaAllDistributions	53	Fits to all continuous and discrete distributions (except Yes-No)

Note: The cbDfaExtremeValue constant is deprecated. It is included for Crystal Ball 2000.5 (5.5) compatibility and is replaced by cbDfaMinExtreme and cbDfaMaxExtreme in later versions of Crystal Ball . Beginning with version 11.1.1.0.00, the lognormal distribution has three parameters instead of two. If this call is used in a model that was created in an earlier version of Crystal Ball, the third parameter has a value of 0.

The following table shows *Criterion* parameter values, used to specify which fitting statistics to return.

Table 78 CB.Fit Criterion Parameter Values — Required, Integer

Named Constant Value	Index Value	Description
cbFitChiSquare	1	Returns the chi-square statistic
cbFitChiSquareP	2	Returns the chi-square statistic *p*-value.
cbFitKolmogorovSmirnov	3	Returns the Kolmogorov-Smirnov statistic
cbFitAndersonDarling	4	Returns the Anderson-Darling statistic
cbDfaAutoSelectRanking Method	5	Automatically selects a ranking method depending on characteristics of the data

CB.Fit Example 1

This example selects a range of data in column D and then calls CB.Fit in a For loop. The loop fits the triangular distribution to the range of data and prints the results of each possible test to the spreadsheet (next to the original data) in cells E4 to E7.

```
CB.SetFitRange Range("D4:D104")
For i = 1 to 4
```

```
   Range("E3").Offset(i).Value = CB.Fit(cbDfaTriangular, i)
Next i
```

CB.Fit Example 2

This example selects a range of data in column D and then calls CB.Fit in a For loop. The loop fits each distribution to the range of data, saving the best distribution index in bestdist and the corresponding Anderson-Darling p-value in bestfit.

```
CB.SetFitRange Range("D4:D104")
For i = 0 To 23
   Result = CB.Fit(i, cbFitAndersonDarling, True)
   If i = 0 Then
      bestfit = result
   Else
      If bestfit > result Then
         bestdist = i
         bestfit = result
      End If
   End If
Next i
```

CB.Fit Example 3

This example automatically selects distributions and ranking methods, and reports the best fit.

```
CB.SetFitRange ("A1:A1000")
Dim DistBestFit As Integer
Dim CritBestFit As Integer
Result = cb.Fit(cbDfaAutoSelectDist, cbDfaAutoSelectRankingMethod, False, _
DistBestFit, CritBestFit)
' Write results (the Goodness-Of-Fit stat and the best fitted distribution)
Sheet1.Cells(4, 4).Value = Result
Sheet1.Cells(5, 4).Value = DistBestFit
Sheet1.Cells(6, 4).Value = CritBestFit
```

CB.Fit Example 4

This example fits a dataset to a gamma distribution while locking the location and shape parameter.

```
' First fit with locking set but switched off
' Location is locked to 0, and shape to 3
CB.SetFitRange ("A1:A30")
' Fit to a gamma distribution
' Switch off all the locked parameters
CB.LockFitParm cbDfaGamma, False
' Report the locked parameter values
Sheet1.Cells(6, 12).Value = cb.GetLockFitParm(cbDfaGamma, 1)     ' Location
Sheet1.Cells(8, 12).Value = cb.GetLockFitParm(cbDfaGamma, 3)     ' Shape
' Get result
Result = CB.Fit(cbDfaGamma, cbFitAndersonDarling)
' Write the parameters and results
```

```
Sheet1.Cells(6, 9).Value = cb.GetFitParm(1)
Sheet1.Cells(7, 9).Value = cb.GetFitParm(2)
Sheet1.Cells(8, 9).Value = cb.GetFitParm(3)
Sheet1.Cells(9, 9).Value = Result

' Now perform fit with locking
' First lock only the location parameter
CB.LockFitParm cbDfaGamma, True, 1, 0
' Shape is not locked (we could have omitted this line)
CB.LockFitParm cbDfaGamma, True, 3, cbParmUndefined
' Get fit result
Result = CB.Fit(cbDfaGamma, cbFitAndersonDarling, False)
' Write the parameters and results
Sheet1.Cells(6, 10).Value = cb.GetFitParm(1)
Sheet1.Cells(7, 10).Value = cb.GetFitParm(2)
Sheet1.Cells(8, 10).Value = cb.GetFitParm(3)
Sheet1.Cells(9, 10).Value = Result

' Now lock the shape as well
CB.LockFitParm cbDfaGamma, True, 3, 3
' Get fit result
Result = CB.Fit(cbDfaGamma, cbFitAndersonDarling, False)
' Write the parameters and results
Sheet1.Cells(6, 11).Value = cb.GetFitParm(1)
Sheet1.Cells(7, 11).Value = cb.GetFitParm(2)
Sheet1.Cells(8, 11).Value = cb.GetFitParm(3)
Sheet1.Cells(9, 11).Value = Result
```

CB.Fit Related Calls

Name	Description
CB.DefineAssum	Defines an assumption in the selected cell using a dialog
CB.DefineAssumND	Defines an assumption in the selected cell without using a dialog
CB.GetFitParm	Returns the value of the specified distribution parameter for the last CB.Fit call
CB.GetLockFitParm	Returns whether a distribution and, optionally, specific parameters have locked values for fitting purposes
CB.LockFitParm	Sets parameter locking for distributions and, optionally, specific parameters
CB.SetFitRange	Specifies the data range to use as input for CB.Fit

CB.FormatPrefs

This subroutine opens the Axis tab of the Chart Preferences dialog so you can change the axis label preferences, axis scale, and number formatting for the selected forecast.

Note: Before calling this subroutine to change the axis label preferences for a particular forecast chart, select the corresponding forecast cell.

CB.FormatPrefs Example

This example runs a simulation of 500 trials, selects a forecast cell (B7), and then opens the Format Preferences dialog for the forecast in B7.

```
CB.Simulation 500
Range("B7").Select
CB.FormatPrefs
```

CB.FormatPrefs Related Calls

Name	Description
CB.ChartPrefs	Sets chart preferences for the selected forecast or assumption using a dialog.
CB.ChartPrefsND	Sets chart preferences for the selected forecast or assumption without a dialog
CB.FormatPrefsND	Sets format preferences for the selected forecast without a dialog

CB.FormatPrefsND

This subroutine sets format preferences for the forecast corresponding to the selected cell without using the Chart Preferences dialog. The defaults are the current settings on the Axis tab of the Chart Preferences dialog.

Select a forecast cell before calling this subroutine to change the format preferences for a single forecast. If you call this subroutine to change a single forecast chart (*All* is False) when no forecast cell is selected, CB.MacroResult returns cbErrNoForecastInCell.

See the *Oracle Crystal Ball User's Guide* for more information on how to format forecast charts.

Table 79 CB.FormatPrefsND Parameters

Parameter	VBA Data Type	Value	Description
Index	Integer	See Table 80 for values	Specifes which forecast chart Axis preference to change. Values appear in Table 80: 1 = cbFmtFormatType, 2 = cbFmtUnderlying, 3 = cbFmtCommas, 4 = cbFmtDecimalPlace.
Value	Variant	See Table 81 for values	Used with the *Index* parameter to specify forecast chart Axis preference settings. Values appear in Table 81.
All (Optional)	Variant	True or False	True applies this option to all forecasts. False applies this option only to the selected forecast. The default is False.

The *Index* parameter (Table 80 on page 133) works with the *Value* parameter to set Axis preferences for forecast charts.

Table 80 CB.FormatPrefsND Index parameter values — Required, Integer

Named Constant Value	Index Value	Description
cbFmtFormatType	1	Sets the number format of the axis labels, according to the *Value* parameter setting
cbFmtUnderlying	2	Sets whether to use the underlying cell format for the axis labels, according to the *Value* parameter setting. You cannot use any other format preferences unless this option is turned off.
cbFmtCommas	3	Sets whether to use commas (or another thousands separator character indicated in the Windows regional settings) in the axis labels, according to the *Value* parameter setting. This preference is ignored for scientific notation.
cbFmtDecimalPlace	4	Sets the number of decimal places to use in the axis labels for fixed and scientific number formats, according to the *Value* parameter setting

The *Value* parameter (Table 81) works with the Index parameter to set forecast chart Axis preferences.

Table 81 CB.FormatPrefsND Value Parameter Values — Required, Variant

Used With Specified Values of Index	Named Constant or Boolean Value	Index Value	Description
For *Index* = 1, cbFmtFormatType: Integer	cbFttFixed	1	Sets the axis labels to have a fixed decimal place format
For *Index* = 1, cbFmtFormatType: Integer	cbFttGeneral	2	Sets the axis labels to be a general Microsoft Excel number format
For *Index* = 1, cbFmtFormatType: Integer	cbFttScientific	3	Sets the axis labels to appear in scientific notation
For *Index* =2, cbFmtUnderlying: Boolean	True or False	n/a	True uses the underlying cell format, overriding any other formatting previously set; False uses General formatting
For *Index* =3, cbFmtCommas: Boolean	True or False	n/a	True uses commas or another thousands separator character in the axis labels; False does not
For *Index* =4, cbFmtDecimalPlace: Integer	Any whole number up to 232	n/a	Sets the number of fixed decimal places. This works only if you set the format type to cbFttFixed or cbFttScientific.

CB.FormatPrefsND Example 1

This example selects a forecast cell, turns off the underlying cell format preference in case it is on, and then changes the format of the axis labels to be a fixed number of decimal places: 3.

```
Range("B7").Select
cb.FormatPrefsND cbFmtUnderlying, False
cb.FormatPrefsND cbFmtFormatType, cbFttFixed
cb.FormatPrefsND cbFmtDecimalPlace, 3
```

CB.FormatPrefsND Example 2

This example sets the format of the axis labels for all the forecasts to use the underlying cell format. This overrides any other format preference until you turn it off.

```
CB.FormatPrefsND cbFmtUnderlying, True, True
```

CB.FormatPrefsND Related Calls

Name	Description
CB.ChartPrefs	Sets chart preferences for the selected forecast or assumption using a dialog
CB.ChartPrefsND	Sets chart preferences for the selected forecast or assumption without a dialog
CB.FormatPrefs	Sets format preferences for the selected forecast using a dialog

CB.Freeze

This subroutine displays a dialog so you can select which assumptions, decision variables, or forecasts will hold their worksheet values during simulations.

Note: Before calling this subroutine, activate a worksheet and reset the simulation.

CB.Freeze Example

This example activates Futura with OptQuest.xls, resets any previously run simulation, and opens the Freeze dialog.

```
Workbooks("Futura with OptQuest.xls").Worksheets("Model").Activate
CB.ResetND
CB.Freeze
```

CB.Freeze Related Calls

Name	Description
CB.FreezeND	Defines which assumptions, decision variables, or forecasts will hold their worksheet values during a simulation, without using a dialog
CB.Reset	Resets a simulation after prompting
CB.ResetND	Resets a simulation without confirmation

CB.FreezeND

This subroutine defines which assumptions, decision variables, or forecasts will hold their worksheet values during a simulation.

Tip: Before calling this subroutine, activate a worksheet and reset the simulation. You must have at least one assumption, decision variable, or forecast in the list of cells to freeze oe enable. Otherwise, CB.MacroResult returns an error.

To freeze cells, build a list of cells to be "frozen" and then use the cbFrzSetList constant to actually freeze the list.

To unfreeze cells and freeze different ones, add all frozen cells to the list of frozen cells, unfreeze them, and then clear the list before freezing new cells.

Note: It is not possible to select a range of cells in the format Range("D6:E527") and add that to the list of frozen cells. You must select and add each cell individually as shown in "CB.FreezeND Example" on page 136. To freeze a large number of cells, the most efficient way is to enumerate through all of the assumptions, decision variables, and forecasts on the sheet you wish to remove from the simulation and then freeze the Crystal Ball data in those cells one at a time. You should be able to refer to the example code for CB.EnumAssum, for example, to see how the enumeration should work. When you are done with the simulation, you will need to unfreeze the Crystal Ball data in the same way.

Table 82 CB.FreezeND Parameters

Parameter	VBA Data Type	Value	Description
Index	Integer	See Table 83 for Index values	Defines which assumptions, decision variables, or forecasts should be frozen (hold to their worksheet values during a simulation). Values, listed in Table 83, are: 1 = cbFrzAddCell, 2 = cbFrzClearlist, 3 = cbFrzSetList.
Frozen (Optional)	Integer	True or False	Sets whether the cells in the list (created by the Index parameter) are frozen during simulations (True) or enabled while all unlisted cells are frozen (False). The default is True.

The following table shows named constant and index values for the *Index* parameter.

Table 83 CB.FreezeND Index Parameter Values — Required, Integer

Named Constant Value	Index Value	Description
cbFrzAddCell	1	Adds the selected cell to the list of frozen or enabled cells
cbFrzClearlist	2	Clears the list of frozen or enabled cells added with cbFrzAddCell

Named Constant Value	Index Value	Description
cbFrzSetList	3	Depending on the value of the *Frozen* parameter, indicates that the listed cells are frozen (True) or the reverse, enabled (False). The default is False. This constant must be called after cbFrzAddCell to activate the freeze. (See CB.FreezeND Example, following.)

CB.FreezeND Example

This example resets the model, adds cells B6 and B7 to the list of frozen (enabled) cells, clears those cells from the list along with any others added previously, selects cell C5, and then adds it to the list of frozen cells. Then, the last line sets this list to be the list of cells to freeze during any simulations. Cell C5 is the only cell included in the final list.

```
CB.ResetND
Range("B6").Select
CB.FreezeND cbFrzAddCell
Range("B7").Select
CB.FreezeND cbFrzAddCell
CB.FreezeND cbFrzClearList
Range("C5").Select
CB.FreezeND cbFrzAddCell
CB.FreezeND cbFrzSetList, True
```

CB.FreezeND Related Calls

Name	Description
CB.Freeze	Defines which assumptions, decision variables, or forecasts to hold to their worksheet values during a simulation, using a dialog
CB.Reset	Resets a simulation after confirmation prompting
CB.ResetND	Resets a simulation without confirmation prompting

CB.GetAssum

This function retrieves information for a specific assumption cell. You can call this function from either a subroutine or a worksheet.

Note: To call this function from a worksheet, use the name CB.GetAssumFN. Named constants do not work as arguments in worksheets. You must use the index number instead.

Note that the CB.GetAssum call is not retrieving the name and value of the third and fourth parameters properly for assumptions defined with alternate parameter sets of a beta distribution via CB.DefineAltParms.

For best results, call CB.CheckData before calling this function to ensure the latest data values are used.

Table 84 CB.GetAssum Returned Data Type

Returned Value	Returned Data Type
Some attribute or parameter for the assumption	Variant

Table 85 CB.GetAssum Parameters

Parameter	VBA Data Type	Value	Description
AssumReference	Variant	In a macro: Range("B3") In a worksheet: B3	Points to the desired assumption cell
Index	Integer	See Table 86	Retrieves information for a specific assumption cell using constant and index values listed in Table 86
ParmNumber (Optional)	Variant	1, 2, 3, 4	Specifies the target distribution parameter for *Index* equal to either cbAsmParmName or cbAsmParmValue

Note: For hypergeometric distributions, whether the distribution was created in pre-7.0 releases with the Probability parameter or in later versions of Crystal Ball with the Success parameter, index 17 for cbDfaHypergeometricSuccess is always returned and the first parameter and value are *Success*, not *Probability*.

Table 86, following, shows named constant and index values for the *Index* parameter.

Note: *Index* values cbAsmNumParms, cbAsmParmName, cbAsmParmValue, cbAsmLowCutOff, and cbAsmHighCutOff are not supported for custom distributions.

Table 86 CB.GetAssum Index Parameter Values — Required, Integer

Named Constant Value	Index Value	Description
cbAsmName	1	Returns the assumption name
cbAsmDistType	2	Returns the index value of the distribution type (See Table 53)
cbAsmNumParms	3	Returns the number of distribution parameters
cbAsmParmName	4	Returns the name of the distribution parameter indexed by *ParmNumber*
cbAsmParmValue	5	Returns the value of the distribution parameter indexed by *ParmNumber*
cbAsmLowCutOff	6	Returns the value of the lower truncation point
cbAsmHighCutOff	7	Returns the value of the higher truncation point
cbAsmDistFlag	8	Returns the extreme value distribution flag: 0=cbFlgMinimum, 1=cbFlgMaximum

Named Constant Value	Index Value	Description
cbAsmFrozen	10	Returns True if the assumption is frozen and False if it is not frozen
cbAsmIsSequential	11	Returns True if the assumption uses sequential sampling and False if it does not

CB.GetAssum Example 1

This example puts information for the first three *Index* constants for the assumption in cell B3 into cells F1 through F3.

In a macro, you can enter:

```
For i = 1 To 3
   CB.CheckData
   Worksheets("sheet3").Range("F1").Offset(i).Value = _
      CB.GetAssum(Worksheets("sheet1").Range("B3"), i)
Next i
```

CB.GetAssum Example 2

This example enters the name of the assumption in cell B3 into the cell that holds this formula.

In a worksheet cell, you can enter:

```
=CB.GetAssumFN(B3,1)
```

CB.GetAssum Related Calls

Name	Description
CB.DefineAssumND	Defines or changes assumptions in selected cells without a dialog
CB.EnumAssum	Enumerates assumption cells for open worksheets
CB.GetAssumPercent	Returns percentile for an assumption cell
CB.GetDecVar	Retrieves information for a specific decision variable cell
CB.GetFore	Retrieves information for a specific forecast cell

CB.GetAssumPercent

This function calculates the value of the specified assumption that corresponds to the specified percentile. You can call this function from either a subroutine or a worksheet. To call this function from a worksheet, use the name CB.GetAssumPercentFN.

Note: In VBA code, always call CB.CheckData before calling this function to ensure the latest data values are used.

Table 87 CB.GetAssumPercent Returned Data Type

Returned Value	Returned Data Type
Assumption value that corresponds to the specified percentile	Variant

The percentile is calculated based on the assumption's distribution. Any truncation points are factored into the percentile calculation.

Note: Reversing the percentiles using the cbRunReversePercentile option does not affect the output of this function. For more information, see "CB.RunPrefsND" on page 187.

Table 88 CB.GetAssumPercent Parameters

Parameter	VBA Data Type	Value	Description
AssumReference	Variant	In a macro: Range("B3") In a worksheet: B3	Points to the desired assumption cell
Percent	Double	A number between 0 and 100	Specifies the percentile for which you want the assumption value

CB.GetAssumPercent Example 1

This example runs a simulation of 1000 trials and then puts the values for every fifth percentile into an array called Percents.

In a macro, you can enter:

```
CB.Simulation 1000
Dim Percents(19) As Double
For i=1 To 19
   CB.CheckData
   Percents(i) = CB.GetAssumPercent (Range("C9"),i*5)
Next i
```

CB.GetAssumPercent Example 2

This example enters the value at the 50th percentile of the assumption distribution in cell B3.

In a worksheet cell, you can enter:

```
=CB.GetAssumPercentFN(B3,50)
```

CB.GetAssumPercent Related Calls

Name	Description
CB.EnumAssum	Enumerates assumption cells for all open workbooks
CB.GetAssum	Retrieves information for a specific assumption cell
CB.GetForePercent	Returns percentiles for forecast cells

CB.GetBatchFitOption

This function returns specified settings made by CB.BatchFitND or the Batch Fit tool.

Table 89 CB.GetBatchFitOption Returned Data Type

Returned Value	Returned Data Type
The specified Batch Fit settings	Variant

Table 90 CB.GetBatchFitOption Parameters — Required, Integer

Parameter	Required?	VBA Data Type	Table
Index	Required	Integer	Table 91
Value1	Optional	Variant	Table 92
Value2	Optional	Variant	Table 93

Table 91 CB.GetBatchFitOption Index Parameter — Required, Integer

Named Constant Value	Index Value	Description
cbBftInputRange	1	Returns a string with the address of the selected Batch Fit data range in the format: '[workbook]sheet'!$A$1:$Z$100
cbBftInputOrientation	2	Returns a constant to indicate whether the data cells are in rows or columns: cbBftInputRows [1] or cbBftInputColumns [2]
cbBftInputHeader	3	Returns a boolean to indicate whether the input range has a header (True)
cbBftInputLabel	4	Returns a boolean to indicate whether the input range has a label (True)
cbBftSelectDist	5	Returns an array of cbDfa... assumption distribution integers as listed in Table 53
cbBftRankMethod	11	Returns the constant of the selected fit ranking method as listed in Table 78

Named Constant Value	Index Value	Description
cbBftOutputLocation	12	Returns the constant of the selected output location setting, a new workbook, the current workbook, a new worksheet, or the current worksheet: cbBftNewWorkbook [1], cbBftCurrentWorkbook [2], cbBftNewWorksheet [3], or cbBftCurrentWorksheet [4] If cbBftCurrentWorksheet is returned, you can use cbBftOutputLocationCell [25] to return the starting cell.
cbBftOutputFormat	13	Returns a boolean to indicate whether the output is formatted (True)
cbBftOutputCorrelationMatrix	14	Returns a boolean to indicate whether a correlation matrix is produced (True)
cbBftOutputCorrelation	15	Returns a boolean to indicate whether assumptions greater than a certain absolute value threshold are correlated (True); if True, use cbBftOutputCorrelationThreshold [26] to return the threshold value
cbBftLinkToCorrelationMatrix	16	Returns a boolean to indicate whether there is a link to the matrix of correlations between the data series (True), assuming cbBftOutputCorrelationMatrix [14] is set to True
cbBftOutputGoodnessOfFitReport	17	Returns a boolean to indicate whether a goodness-of-fit report is produced (True); if True, use cbBftOutputGoodnessOfFitSheet [27] to return the sheet name of the report
cbBftOutputAssumReport	18	Returns a boolean to indicate whether an assumption report is produced (True); if True, use cbBftOutputAssumSheet [28] to return the sheet name of the report
cbBftOutputAssumFullStatistics	19	If cbBftOutputAssumReport [18] is True, can be used to return a boolean to indicate whether the assumption report includes full statistics (True)
cbBftUseDistLocking	21	Returns a boolean to indicate whether parameter locking is switched on (True)
cbBftOutputGoodnessOfFitAllStats	22	Returns a boolean to indicate whether all statistics should appear when a Goodness of Fit report is produced (True)
cbBftFitDistLockParam	23	Returns the locked value of the specified distribution and parameter. The following distributions from Table 53 can be used as Value1: ● Gamma distribution – Location, Shape parameters ● Lognormal distribution – Location parameter ● Student's *t* distribution – Degrees of Freedom parameter ● Weibull distribution – Location, Shape parameters ● Binomial distribution – Trials parameter ● Hypergeometric distribution – Trials parameter Use Value2 to specify the number of the target parameter.
cbBftOutputLocationCell	25	If cbBftOutputLocation [12] returns cbBftCurrentWorksheet [4], can be used to return the first cell of the output range (in the upper left corner)
cbBftOutputCorrelationThreshold	26	If cbBftOutputCorrelation [15] returns True, can be used to return the correlation threshold absolute value (default = 0.0)
cbBftOutputGoodnessOfFitSheet	27	If cbBftOutputGoodnessOfFitReport [17] returns True, can be used to return the sheet name string of the report

Named Constant Value	Index Value	Description
cbBftOutputAssumSheet	28	If cbBftOutputAssumReport [18] returns True, can be used to return the sheet name string of the report
cbBftOutputWorksheet	29	Returns whether the current sheet (cbBftCurrentWorksheet [4]) or new sheet (cbBftNewWorksheet [3]) is being used for the Batch Fit output
cbBftOutputOrientation	30	Returns a constant to indicate whether the output is in rows or columns: cbBftOutputRows [1] or cbBftOutputColumns [2]
cbBftOutputSheetName	31	Returns the name of the worksheet containing primary fit results (assumptions) as a string

Table 92 CB.GetBatchFitOption Value1 Parameter — Optional, Variant

Used With Specified Values of Index	Named Constant or Boolean Value	Index Value	Description
For *Index* = 23, cbBftFitDistLockParam: Variant	One of the cbDfa[*distribution*]... names for the following distributions from Table 53: gamma, lognormal, Student's *t*, Weibull, binomial, hypergeometric.	n/a	Returns the locked value of the specified distribution and the parameter specified with *Value2*

Table 93 CB.GetBatchFitOption Value2 Parameter — Optional, Variant

Used With Specified Values of Index	Named Constant or Boolean Value	Index Value	Description
For *Index* = 23, cbBftFitDistLockParam: Integer	One of the parameter numbers from Table 53 for the distribution specified with Value1: gamma, lognormal, Student's *t*, Weibull, binomial, hypergeometric.	n/a	Returns the locked value of the specified parameter for the distribution specified with *Value1*. The numbers listed for the following parameters in Table 53 can be used for *Value2*: ● Gamma distribution — Location, Shape parameters ● Lognormal distribution — Location parameter ● Student's *t* distribution — Degrees of Freedom parameter ● Weibull distribution — Location, Shape parameters ● Binomial distribution — Trials parameter ● Hypergeometric distribution — Trials parameter

CB.GetBatchFitOption Example 1

This example returns the address of the selected input data range as a string.

```
CB.GetBatchFitOption cbBftInputRange
```

CB.GetBatchFitOption Example 2

This example returns the locked value for parameter 3, Location, for the lognormal distribution.

```
CB.GetBatchFitOption cbBftFitDistLockParam, cbDfaLogNormal, 3
```

CB.GetBatchFitOption Related Calls

Name	Description
CB.BatchFit	Launches the Batch Fit tool
CB.BatchFitND	Runs the Batch Fit tool without displaying dialogs

CB.GetBootstrapOption

This function returns specified settings made by CB.BootstrapND or the Bootstrap tool.

Table 94 CB.GetBootstrapOption Returned Data Type

Returned Value	Returned Data Type
The specified Bootstrap settings.	Variant

Table 95 CB.GetBootstrapOption Parameters

Parameter	Required?	VBA Data Type	Table
Index	Required	Integer	Table 96
Index2	Optional	Integer	Table 97
Value	Optional	Variant	Table 98

Table 96 CB.GetBootstrapOption Index Parameter – Required, Integer

Named Constant Value	Index Value	Description
cbBtsGetTarget	1	Returns a string with the address of the target forecast in the format: '[workbook]sheet'!$A$1
cbBtsGetDataRange	2	Returns a string with the address of the selected bootstrap data range in the format:'[workbook]sheet'!$A$1:$Z$100
cbBtsMethod	3	Returns the constant of the selected bootstrap method: cbBtsMethodResample [1] or cbBtsMethodMultiSim [2].
cbBtsAnalyze	4	Returns the constant of the selected analysis setting: cbBtsAnalyzeStat [1], cbBtsAnalyzePercent [2], or cbBtsAnalyzeCapMetrics [3].
cbBtsPercent	5	Returns the constant of the currently selected percentile setting (for use if cbBtsAnalyze is called with cbBtsAnalyzePercent [2]): cbBtsPercentTenNinety [1], cbBtsPercentFiveNinetyFive [2], cbBtsPercentQuartiles [3], cbBtsPercentQuintiles [4], cbBtsPercentDeciles [5], cbBtsPercentIcosatiles [6], or cbBtsPercentCustom [7]. If the returned value is cbBtsPercentCustom [7], you can use that constant as an *Index2* parameter to return a range of custom values.
cbBtsSample	6	Used with *Index2* to return the current bootstrap sample settings for cbBtsNumberSamples [1] or cbBtsTrialsPerSample [2]

Named Constant Value	Index Value	Description
cbBtsForeOption	7	Returns the constant of the current forecast display setting: cbBtsShowDefinedFore [1], cbBtsShowTargetFore [2], or cbBtsHideFore [3]

Table 97 CB.GetBootstrapOption Index2 Parameter – Optional, Integer

Related Value	Named Constant Value	Index	Description
For *Index* = 5, cbBtsPercent: Integer	cbBtsPercentCustom	7	Returns a range of custom percentile values in the range specified with *Value*
For *Index* = 6, cbBtsSample: Integer	cbBtsNumberSamples	1	Used with cbBtsSample to return the number of bootstrap samples to use
For *Index* = 6, cbBtsSample: Integer	cbBtsTrialsPerSample	2	Used with cbBtsSample to return the number of trials to run for each bootstrap sample

Table 98 CB.GetBootstrapOption Value Parameter – Optional, Variant

Related Value	VBA Data Type	Description
For *Index2* = 7, cbBtsPercentCustom: Variant	A string containing a Microsoft Excel range to holds a set of returned custom percentile values	Used with cbBtsPercent and cbBtsPercentCustom to return custom percentile values and place them within the specified range

CB.GetBootstrapOption Example 1

This example returns the percentile setting used in the Bootstrap tool definition. For this example, the returned type would be cbBtsPercentCustom.

```
CB.GetBootstrapOption cbBtsPercent
```

CB.GetBootstrapOption Example 2

This example returns a set of custom percentile values (for example, those stored in the Microsoft Excel range F17:H17, as shown in CB.BootstrapND Example 1). The returned values are placed in the specified range, in this case, F20:H20.

```
CB.GetBootstrapOption cbBtsPercent, cbBtsPercentCustom, "F20:H20"
```

CB.GetBootstrapOption Related Calls

Name	Description
CB.Bootstrap	Launches the Bootstrap tool
CB.BootstrapND	Runs the Bootstrap tool without displaying dialogs

CB.GetCBAutoLoad

In Crystal Ball 7.3.*x* or 11.*x*, you can use the following alternatives to determine whether Crystal Ball launches each time Microsoft Excel starts, called "autoloading:"

- Start the Crystal Ball Application Manager and check or uncheck **When starting Microsoft Excel, automatically launch Crystal Ball**.
- Open the Microsoft Excel COM Add-ins dialog and check or uncheck **Crystal Ball**.
- Include the CB.SetCBAutoLoad call ("CB.SetCBAutoLoad" on page 207) in Visual Basic or VBA code with the autoLoad parameter set to True or False.

Whichever alternative you use, the same setting appears in both of the other two alternatives.

For more information on these alternatives, see "Starting Crystal Ball " in the current *Oracle Crystal Ball Installation and Licensing Guide*.

CB.GetCBAutoLoad returns the current autoload setting, True if Crystal Ball is set to automatically load each time Microsoft Excel opens or False if Crystal Ball is not set to autoload.

Table 99 CB.GetCBAutoLoad Returned Data Type

Returned Value	Returned Data Type
The current autoload setting.	Boolean

CB.GetCBAutoLoad Example

To retrieve the current Crystal Ball autoload setting, use this call in a function in the following form:

```
ReturnValue = CB.GetCBAutoLoad()
```

CB.GetCBAutoLoad Related Calls

Name	Description
CB.CBLoaded	Indicates whether Crystal Ball is currently loaded within Microsoft Excel
CB.SetCBAutoLoad	Sets or cancels autoloading with Microsoft Excel; determines whether Crystal Ball is set to open whenever Microsoft Excel opens
CB.Shutdown	Closes Crystal Ball while leaving Microsoft Excel open
CB.Startup	Loads Crystal Ball into Microsoft Excel if possible and indicates when Crystal Ball has been successfully opened

CB.GetCertainty

This function calculates the certainty level (or probability, as a percent) of achieving a forecast value at or smaller than a specific threshold. The value returned is equivalent to setting the right

(highest value) endpoint grabber on the forecast chart to a specific value and obtaining the calculated certainty level. You can call this function from either a macro or a worksheet.

For example, suppose trial values for a forecast range from 5 to 300. If you enter 200 for the Value parameter (see Table 101 on page 146), CB.GetCertainty would return the probability of obtaining a value within the range of 5 to 200, including those end points.

Note: To call this function from a worksheet, use the name CB.GetCertaintyFN.

Table 100 CB.GetCertainty Returned Data Type

Returned Value	Returned Data Type
Certainty (probability as a percent) of achieving a value within a range from the minimum up to and including the specified forecast value.	Variant

You must run a simulation before you call this function, or it returns 0.

Table 101 CB.GetCertainty Parameters

Parameter	VBA Data Type	Value	Description
ForeReference	Variant	In a macro: Range("B3") In a worksheet: B3	Points to the forecast cell for which you want the certainty value
Value	Double	Any number	Specifies the desired value

CB.GetCertainty Example 1

This example runs a simulation of 1000 trials and then pastes the certainty of the forecast in B7 being less than or equal to 15,000 into cell M6.

In a macro, you can enter:

```
CB.Simulation 1000
Range("M6").Value = CB.GetCertainty(Range("B7"), 15000)
```

CB.GetCertainty Example 2

This example, before a simulation, displays #VALUE! in the cell. After a simulation, it displays the certainty of the forecast in B7 being less than or equal to 15,000.

In a worksheet cell, you can enter:

```
=CB.GetCertaintyFN(B7,15000)
```

CB.GetCertainty Related Calls

Name	Description
CB.EnumFore	Enumerates forecast cells for all open workbooks
CB.GetAssumPercent	Returns percentiles for assumption cells
CB.GetForePercent	Returns percentiles for a specific forecast
CB.GetForeStat	Returns statistics for forecast cells
CB.Simulation	Runs a Crystal Ball simulation

CB.GetCorrelation

This function returns the correlation coefficient defined between two assumption cells.

Note: For best results, call CB.CheckData before calling this function to ensure the latest data values are used.

Table 102 CB.GetCorrelation Returned Data Type

Returned Value	Returned Data Type
The correlation coefficient of the two assumptions	Variant

If a correlation between the assumptions wasn't defined, this function returns the value 2.

Table 103 CB.GetCorrelation Parameters

Parameter	VBA Data Type	Value	Description
AssumReference	Variant	In a macro: Range("B3")	Points to one of the correlated assumption cells
Column	Integer	Any number up to 256	Points to the column of the second correlated assumption
Row	Long	Any number up to 64,000	Points to the row of the second correlated assumption

CB.GetCorrelation Example

This example puts the correlation coefficient between G8 and G15 into cell H15.

In a macro, you can enter:

```
CB.CheckData
Range("H15").Value = CB.GetCorrelation(Range("G8"), 7, 15)
```

CB.GetCorrelation Related Calls

Name	Description
CB.CorrelateND	Defines a correlation coefficient between two selected assumptions
CB.GetAssum	Retrieves information for a specific assumption cell

CB.GetDataAnalysisOption

This function returns specified settings made by CB.DataAnalysisND or the Data Analysis tool.

Table 104 CB.GetDataAnalysisOption Returned Data Type

Returned Value	Returned Data Type
The specified Data Analysis settings	Variant

CB.GetDataAnalysisOption has one parameter, *Index*, which is a required integer. See Table 105 for details.

Table 105 CB.GetDataAnalysisOption Index Parameter — Required, Integer

Named Constant Value	VBA Data Type	Index Value	Description
cbDataAnalysisSelectData	Integer	2	Returns a string with the address of the target forecast in the format: '[workbook]sheet'!$A$1
cbDataAnalysisDataOrientation	Integer	3	Returns one of two possible values for the data orientation: cbDataAnalysisDataInRows [1] cbDataAnalysisDataInColumns [2]
cbDataAnalysisSelectHeaders	Integer	4	Returns a boolean value to indicate if column header text is in the selection (True)
cbDataAnalysisSelectLabels	Integer	5	Returns a boolean value to indicate if row label text is in the selection (True)
cbDataAnalysisSelectLSL	Integer	6	Returns the range of values selected as forecast LSLs (lower specification limit) for process capability analysis. The range is a string in the format: '[workbook]sheet'!$A$1
cbDataAnalysisSelectUSL	Integer	7	Returns the range of values selected as forecast USLs (upper specification limit) for process capability analysis. The range is a string in the format: '[workbook]sheet'!$A$1

Named Constant Value	VBA Data Type	Index Value	Description
cbDataAnalysisSelectTarget	Integer	8	Returns the range of values selected as forecast Targets for process capability analysis. The range is a string in the format: '[workbook]sheet'!$A$1
cbDataAnalysisAutoOpenCharts	Integer	9	Returns a boolean value to indicate whether to open forecast charts automatically when the data analysis runs (True)
cbDataAnalysisSetViewType	Integer	10	If charts are shown, returns cbChtChartType constants from CB.ChartPrefsND to indicate which chart view appears. See Table 24 for returned constants.
cbDataAnalysisFitToDistribution	Integer	11	Returns a boolean value to indicate whether forecast charts should be fitted to a distribution (True)
cbDataAnalysisRankingMethod	Integer	12	Returns the ranking method for distribution fitting. See Table 78 for constants
cbDataAnalysisGenerateCorrelationMatrix	Integer	13	Returns a boolean value to indicate whether rank correlations between forecast charts should be displayed in a matrix (True)
cbDataAnalysisRunOnAllModels	Integer	14	Returns a boolean value to indicate whether a simulation is run on all open workbooks at the same time the selected data is analyzed (True)
cbDataAnalysisNumberOfSeries	Integer	15	Returns an integer value equal to the number of currently selected data series
cbDataAnalysisNumberOfPoints	Integer	16	Returns an integer value equal to the number of points in each data series

CB.GetDataAnalysisOption Example 1

The following example returns a string with the address of the selected data range:

```
CB.GetDataAnalysisOption cbDataAnalysisSelectData
```

CB.GetDataAnalysisOption Example 2

The following example returns the current view setting for forecast charts generated by the Data Analysis tool:

```
CB.GetDataAnalysisOption cbDataAnalysisSetViewType
```

CB.GetDataAnalysisOption Related Calls

Name	Description
CB.DataAnalysis	Launches the Data Analysis tool

Name	Description
CB.DataAnalysisND	Runs the Data Analysis tool without displaying dialogs

CB.GetDecisionTableOption

This function returns specified settings made by CB.DecisionTableND or the Decision Table tool.

Table 106 CB.GetDecisionTableOption Returned Data Type

Returned Value	Returned Data Type
The specified Decision Table settings	Variant

Table 107 CB.GetDecisionTableOption Parameters

Parameter	Required?	VBA Data Type	Table
Index	Required	Integer	Table 108
Index2	Optional	Integer	Table 109

Table 108 CB.GetDecisionTableOption Index Parameter – Required, Integer

Named Constant Value	Index Value	Description
cbDecGetTarget	1	Returns a string with the address of the target forecast in the format: '[workbook]sheet'!$A$1
cbDecGetDecVar	2	Returns a string with the address of the selected decision variable in the format: '[workbook]sheet'!$A$1 Must be used with *Index2* (cbDecVar1 or cbDecVar2) to indicate which decision variable's address to return.
cbDecSimOption	3	Used with *Index2* to return the number of test values for the specified decision variable or the maximum number of trials for the simulation
cbDecForeOption	4	Returns the current forecast chart display setting: 1 = Show Forecasts As Defined (set with cbDecShowDefinedFore) 2 = Show Only Target Forecast (set with cbDecShowTargetFore) 3 = Hide All Forecasts (set with cbDecHideFore)

Table 109 CB.GetDecisionTableOption Index2 Parameter – Optional, Integer

Named Constant Value	Index Value	Description
cbDecVar1	1	Used with cbDecGetDecVar to return the address of the first selected decision variable in the format: '[workbook]sheet'!$A$1
cbDecVar2	2	Used with cbDecGetDecVar to return the address of the second selected decision variable in the format: '[workbook]sheet'!$A$1
cbDecNumValuesVar1	1	Used with cbDecSimOption to return the number of test values to use for the first selected decision variable
cbDecNumValuesVar2	2	Used with cbDecSimOption to return the number of test values to use for the second selected decision variable
cbDecSimTrials	3	Used with cbDecSimOption to return the maximum number of trials for each simulation

CB.GetDecisionTableOption Example

The following example returns information for the Decision Table definition created in CB.DecisionTableND Example and stores it in cells M6 through M12.

```
'Gets the address of the target forecast
Range("M6").Value = CB.GetDecisionTableOption(cbDecGetTarget)

'Gets the address of the first and second selected decision variables
Range("M7").Value = CB.GetDecisionTableOption(cbDecGetDecVar, cbDecVar1)
Range("M8").Value = CB.GetDecisionTableOption(cbDecGetDecVar, cbDecVar2)

'Gets the number of test values for the first and second decision variables
Range("M9").Value = CB.GetDecisionTableOption(cbDecSimOption, _
   cbDecNumValuesVar1)
Range("M10").Value = CB.GetDecisionTableOption(cbDecSimOption, _
   cbDecNumValuesVar2)

'Gets the maximum number of trials per simulation
Range("M11").Value = CB.GetDecisionTableOption(cbDecSimOption, _
   cbDecSimTrials)

'Returns the forecast chart window display setting
Range("M12").Value = CB.GetDecisionTableOption(cbDecForeOption)
```

CB.GetDecisionTableOption Related Calls

Name	Description
CB.DecisionTable	Launches the Decision Table tool
CB.DecisionTableND	Runs the Decision Table tool without displaying dialogs

CB.GetDecVar

This function retrieves information for a specific decision variable cell.

Note: For best results, always call CB.CheckData before calling this function to ensure the latest data values are used.

Table 110 CB.GetDecVar Returned Data Type

Returned Value	Returned Data Type
The specified decision variable data	Variant

Table 111 CB.GetDecVar Parameters

Parameter	VBA Data Type	Value	Description
DecVarReference	Variant	In a macro: Range("B3")	Points to the desired decision variable cell
Index	Integer	See Table 112.	Returns specified data from the selected decision variable using the constant and index values shown in Table 112

The following table shows named constant and index values for the *Index* parameter.

Table 112 CB.GetDecVar Index Parameter Values — Required, Integer

Named Constant Value	Index Value	Description
cbDecName	1	Returns the decision variable name
cbDecMin	2	Returns the decision variable minimum (lower) bound
cbDecMax	3	Returns the decision variable maximum (upper) bound
cbDecType	4	Returns the type of decision variable: 0 = Continuous, 1 = Discrete, 2 = Binary, 3 = Category, 4 = Custom.
cbDecStep	5	For discrete variables, this returns the step size
cbDecFrozen	6	Returns whether the decision variable in the selected cell is frozen
cbDecCustomData	7	Returns the custom value list or range if cbDecType = 4, Custom
cbDecCustomDataCount	8	Returns the number of values in the custom value list or range if cbDecType = 4, Custom

CB.GetDecVar Example

This example puts all the decision variable information for the decision variable in cell B3 into cells F2 through F9, assuming that cbDecType = 4, Custom.

In a macro, you can enter:

```
For i = 1 To 8
   CB.CheckData
   Worksheets("sheet3").Range("F1").Offset(i).Value = _
      CB.GetDecVar(Worksheets("sheet1").Range("B3"), i)
Next i
```

CB.GetDecVar Related Calls

Name	Description
CB.DefineDecVar	Defines or changes a decision variable in a selected cell using a dialog
CB.DefineDecVarND	Defines or changes a decision variable in a selected cell without a dialog
CB.EnumDecVar	Enumerates decision variable cells for all open workbooks
CB.SetDecVar	Changes certain attributes for a decision variable

CB.GetExcel2007ForegroundMode

This function is used in Microsoft Excel 2007 to retrieve the current setting for the Normal Speed preference "Bring Microsoft Excel 2007 to the foreground while running Normal speed simulations (faster)." When set to True, that preference lets simulations run at optimal speed when using Normal speed in Microsoft Excel 2007.

Table 113 CB.GetExcel2007ForegroundMode Returned Data Type

Returned Value	Returned Data Type
The current foreground mode Run Preferences setting for Normal speed in Microsoft Excel 2007.	Boolean

CB.GetExcel2007ForegroundMode Example

This example returns the current setting (True or False) for the preference "Bring Microsoft Excel 2007 to the foreground while running Normal speed simulations (faster)" in the Normal Speed options dialog, accessed manually by choosing Normal Speed on the Speed tab of the Run Preferences dialog and then clicking the Options button. When set to False, simulations in Microsoft Excel 2007 might run more slowly than if the preference were set to True (the default).

```
CB.GetExcel2007ForegroundMode()
```

CB.GetExcel2007ForegroundMode Related Calls

Name	Description
CB.SetExcel2007ForegroundMode	Sets the Run Preferences foreground mode setting for Microsoft Excel 2007 in Normal speed

CB.GetFitParm

This function returns the value of the specified distribution parameter for the last CB.Fit.

Table 114 CB.GetFitParm Returned Data Type

Returned Value	Returned Data Type
The distribution parameter for the last CB.Fit, as specified by *Index*	Variant

Note: Before calling this function, call CB.Fit, or the function returns 0.

CB.GetFitParm has one parameter, *Index*, an integer. The *Index* parameter specifies which distribution parameter value to return. Its has only four values: 1, 2, 3, and 4. The following table shows which values are returned for each distribution.

Table 115 CB.GetFitParm Index Parameter Values — Required, Integer

Last Distribution	Index = 1 returns...	Index = 2 returns...	Index = 3 returns...	Index = 4 returns...
Normal	mean	standard deviation	n/a	n/a
Triangular	minimum	likeliest	maximum	n/a
Lognormal	mean	standard deviation	location	n/a
Uniform	minimum	maximum	n/a	n/a
Exponential	rate	n/a	n/a	n/a
Weibull	location	scale	shape	n/a
Beta	alpha	beta	maximum	minimum
Gamma	location	scale	shape	n/a
Logistic	mean	scale	n/a	n/a
Pareto	location	shape	n/a	n/a
Extreme value	likeliest	scale	max (True) or min (False)	n/a
Minimum extreme value	likeliest	scale	n/a	n/a
Maximum extreme value	likeliest	scale	n/a	n/a
Student's *t*	midpoint	scale	degrees of freedom	n/a
BetaPERT	minimum	likeliest	maximum	n/a

Note: The parameters for beta have changed since Crystal Ball 2000.x (5.x). There are now four parameters. Extreme value is deprecated. It is included for Crystal Ball 2000.5 (5.5) compatibility and is replaced by `cbDfaMinExtreme` and `cbDfaMaxExtreme` in later versions of Crystal Ball. Beginning with version 11.1.1.0.00, the lognormal distribution has three parameters instead of two. If this call is used in a model that was created in an earlier version of Crystal Ball, the third parameter has a value of 0.

CB.GetFitParm Example

This example selects a range of data in column D and then calls CB.Fit to fit a triangular distribution to the range of data, printing the results of each possible test to the worksheet (next to the original data) in cells E4 to E7. After the fit, it uses CB.GetFitParm to put the Minimum, Likeliest, and Maximum parameters in cells E8, E9, and E10.

```
CB.SetFitRange Range("D4:D104")
For i = 1 to 4
   Range("E3").Offset(i).Value = CB.Fit(cbDfaTriangular, i)
Next i
For j = 1 to 3
   Range("E7").Offset(j).Value = CB.GetFitParm(j)
Next j
```

CB.GetFitParm Related Calls

Name	Description
CB.DefineAssum	Defines an assumption in the selected cell using a dialog
CB.DefineAssumND	Defines an assumption in the selected cell without using a dialog
CB.Fit	Fits a specific probability distribution to a selected set of data (range of cells)
CB.GetLockFitParm	Returns whether a distribution and, optionally, specific parameters have locked values for fitting purposes
CB.LockFitParm	Sets parameter locking for distributions and, optionally, specific parameters
CB.SetFitRange	Specifies the data range to be used as input for CB.Fit

CB.GetFore

This function retrieves information for a specific forecast cell.

Note: For best results, call CB.CheckData before calling this function to ensure the latest data values are used.

Table 116 CB.GetFore Returned Data Type

Returned Value	Returned Data Type
The specified forecast data	Variant

Table 117 CB.GetFore Parameters

Parameter	VBA Data Type	Value	Description
ForeReference	Variant	In a macro: Range("B3")	Points to the desired forecast cell
Index	Integer	See Table 118	Retrieves information for a specific forecast cell using the values in Table 118

The following table (Table 118) shows values available for the *Index* parameter.

Table 118 CB.GetFore Index Parameter Values — Required, Integer

Named Constant Value	Index Value	Description
cbForName	1	Returns the forecast name
cbForUnits	2	Returns the forecast units
cbForAutoOpen	4	Returns whether Crystal Ball displays the forecast window automatically: 0 = no, 1 = yes
cbForOpenWhen	5	Returns when Crystal Ball displays the forecast window: 0 = when running (cbOpnWhenRunning), 1 = when stopped (cbOpnWhenStopped)
cbForPrecision	6	Returns whether Crystal Ball uses precision control: 0 = no, 1 = yes
cbForPrecisionAbsolute	7	Returns what kind of terms Crystal Ball uses to define the confidence interval around the stopping criterion: 0 = relative, 1 = absolute
cbForPrecisionAbsoluteValue	8	Returns the size of the confidence interval in absolute forecast units
cbForPrecisionRelative Value	9	Returns the size of the confidence interval in relative percentage terms
cbForPrecisionMean	10	Returns whether Crystal Ball uses the precision of the mean as a simulation stopping criterion:0 = no, 1 = yes
cbForPrecisionStdDev	11	Returns whether Crystal Ball uses the precision of the standard deviation as a stopping criterion for the simulation: 0 = no, 1 = yes
cbForPrecisionPerc	12	Returns whether Crystal Ball uses the precision of a percentile value as a stopping criterion for the simulation: 0 = no, 1 = yes
cbForPrecisionPercValue	13	Returns the percentile value used to check for the indicated precision
cbForFilter	14	Returns whether Crystal Ball uses forecast filtering: 0 = no, 1 = yes
cbForFilterInside	15	Returns which forecast values Crystal Ball discards in filtering: 0 = the values outside the filtering range, 1 = the values inside the range
cbForFilterFrom	16	Returns the lower bound of the filtering range

Named Constant Value	Index Value	Description
cbForFilterTo	17	Returns the upper bound of the filtering range
cbForFilterGlobal	18	Returns whether global filtering is set for the forecast (True) or not (False)
cbForFrozen	19	Returns whether the forecast cell is frozen
cbForLSL	20	Returns a lower specification limit (LSL) for the forecast
cbForUSL	21	Returns an upper specification limit (USL) for the forecast
cbForTarget	22	Returns a target value for the forecast

Note: The third constant for Index, cbForWindowSize, is obsolete and has been removed from Crystal Ball 7.*x* and 11.*x*.

CB.GetFore Example

This example puts information for *Index* constants 1 through 5 for the forecast in cell B3 into cells F1 through F5.

In a macro, you can enter:

```
For i = 1 To 5
   CB.CheckData
   Worksheets("sheet3").Range("F1").Offset(i).Value = _
      CB.GetFore(Worksheets("sheet1").Range("B3"), i)
Next i
```

Notice that a message appears to indicate the third constant is obsolete and cell F3 is left blank.

CB.GetFore Related Calls

Name	Description
CB.EnumFore	Enumerates forecast cells for all open workbooks
CB.GetAssum	Gets assumption cell information
CB.GetDecVar	Retrieves information for a specific decision variable cell
CB.GetForePercent	Gets a forecast cell percentile
CB.GetForeStat	Gets a forecast cell statistic
CB.SetFore	Changes forecast attributes without redefining the forecast

CB.GetForeData

This function returns the value for the given trial for a specific forecast cell.

Note: To call this function from a worksheet, use the name CB.GetForeDataFN.

Table 119 CB.GetForeData Returned Data Type

Returned Value	Returned Data Type
The value of the given trial for the specified forecast cell	Variant

Table 120 CB.GetForeData Parameters

Parameter	VBA Data Type	Value	Description
ForeReference	Variant	In a macro: Range("B3") In a worksheet: B3	Points to the desired forecast cell
Index	Long	A whole number from 1 through the number of trials in the simulation	Number of the trial whose value you want to return

CB.GetForeData Example 1

This example runs a simulation of 1000 trials and then pastes the value of the 100th trial of the forecast in B1 into cell A10.

In a macro, you can enter:

```
CB.Simulation 1000
Range("A10").Value = CB.GetForeData(Range("B1"), 100)
```

CB.GetForeData Example 2

This example pastes the value of the 100th trial of the forecast in B1 into the cell where the function is entered. In a worksheet, you can enter:

```
=CB.GetForeDataFN(B1,100)
```

CB.GetForePercent

This function calculates the value of the specified forecast that corresponds to the specified percentile. You can call this function from either a macro or a worksheet.

Note: To call this function from a worksheet, use the name CB.GetForePercentFN.

Table 121 CB.GetForePercent Returned Data Type

Returned Value	Returned Data Type
Forecast value that corresponds to the specified percentile	Variant

You must run a simulation before you call this function, or it returns 0.

Note: Reversing the percentiles using the cbRunReversePercentile constant does not affect the output of this function. For more information, see "CB.RunPrefsND" on page 187.

Table 122 CB.GetForePercent Parameters

Parameter	VBA Data Type	Value	Description
ForeReference	Variant	In a macro: Range("B3") In a worksheet: B3	Points to the forecast cell for which you want the percentile value
Percent	Double	A number between 0 and 100	Specifies the percentile for which you want the forecast value

CB.GetForePercent Example 1

This example runs a simulation of 1000 trials and then puts the values for every fifth percentile into an array called Percents.

In a macro, you can enter:

```
CB.Simulation 1000
Dim Percents(19) As Double
For i=1 To 19
   Percents(i) = CB.GetForePercent (Range("C9"),i*5)
Next i
```

CB.GetForePercent Example 2

This example, before a simulation, displays 0 in the cell. After a simulation, it displays the forecast value at the 15th percentile of the forecast in B7.

In a worksheet cell, you can enter:

```
=CB.GetForePercentFN(B7,15)
```

CB.GetForePercent Related Calls

Name	Description
CB.GetCertainty	Returns the certainty level of achieving a forecast value at or smaller than a specific threshold
CB.EnumFore	Enumerates forecast cells for all open workbooks

Name	Description
CB.GetAssumPercent	Returns percentiles for assumption cells
CB.GetFore	Retrieves information for a specific forecast cell
CB.GetForeStat	Returns statistics for forecast cells

CB.GetForeStat

This function calculates the specified statistic for the forecast in the specified cell. You can call this function from either a macro or a worksheet.

Run a simulation before calling this function, or else it returns 0.

Note: To call this function from a worksheet, use the name CB.GetForeStatFN.

CB.GetForeStatFN only pulls N-1 trials during a recalculation for either a single step or a full simulation, but will reflect full trials at the end of the single step or full simulation since a final recalculation is always performed.

Table 123 CB.GetForeStat Returned Data Type

Returned Value	Returned Data Type
The specified statistic for the forecast.	Variant

Table 124 CB.GetForeStat Parameters

Parameter	VBA Data Type	Value	Description
ForeReference	Variant	In a macro: Range("B3")In a worksheet: B3	Points to the forecast cell for which you want the statistic
Index	Integer	See Table 125.	Calculates the specified statistic for the forecast in the specified cell using the values listed in Table 125

Use the following *Index* named constant and index values (Table 125) to return forecast statistics for the selected cell.

Table 125 CB.GetForeStat Index Parameter Values — Required, Integer

Named Constant Value	Index Value	Description
cbFstTrials	1	Returns the number of trials
cbFstMean	2	Returns the mean
cbFstMedian	3	Returns the median
cbFstMode	4	Returns the mode

Named Constant Value	Index Value	Description
cbFstStdDev	5	Returns the standard deviation
cbFstVariance	6	Returns the variance
cbFstSkewness	7	Returns the skewness
cbFstKurtosis	8	Returns the kurtosis
cbFstCV	9	Returns the coefficient of variability
cbFstRangeMin	10	Returns the range minimum
cbFstRangeMax	11	Returns the range maximum
cbFstRangeWidth	12	Returns the range width
cbFstStdError	13	Returns the standard error
cbFstBaseCase	14	Returns the base case (initial value) of the forecast cell
cbFstCp	50	Returns the Cp for short-term data (or Pp for long-term data)
cbFstCpklower	51	Returns the Cpk-lower for short-term data (or Ppk-lower for long-term data)
cbFstCpkupper	52	Returns the Cpk-upper for short-term data (or Ppk-upper for long-term data)
cbFstCpk	53	Returns the Cpk for short-term data (or Ppk for long-term data)
cbFstCpm	54	Returns the Cpm for short-term data (or Ppm for long-term data)
cbFstZLSL	55	Returns the Z-LSL
cbFstZUSL	56	Returns the Z-USL
cbFstZst	57	Returns the Z-st
cbFstZlt	58	Returns the Z-lt
cbFstpNCbelow	59	Returns the p(N/C)-below LSL
cbFstpNCabove	60	Returns the p(N/C)-above USL
cbFstpNCtotal	61	Returns the p(N/C)-total
cbFstPPMbelow	62	Returns the PPM-below LSL
cbFstPPMabove	63	Returns the PPM-above USL
cbFstPPMtotal	64	Returns the PPM-total

Note: To use CB.GetForeStat in a worksheet, use the integer value of the *Index* parameter. The named constant will not work. If no mode is calculated, *Index* = cbFstMode returns #VALUE.

CB.GetForeStat Example 1

This example runs a simulation of 1000 trials and then pastes the standard deviation of the forecast in B7 into cell M6.

In a macro, you can enter:

```
CB.Simulation 1000
Range("M6").Value = CB.GetForeStat(Range("B7"), cbFstStdDev)
```

CB.GetForeStat Example 2

This example, before a simulation, displays 0 in the cell. After a simulation, it displays the standard deviation of the forecast in B7.

In a worksheet cell, you can enter:

```
=CB.GetForeStatFN(B7,5)
```

Note: You must use the numeric value of the Index parameter in a worksheet cell. The named constant will not work.

CB.GetForeStat Related Calls

Name	Description
CB.EnumFore	Enumerates forecast cells for open worksheets
CB.GetCertainty	Returns the certainty level of achieving a forecast value at or smaller than a specific threshold
CB.GetFore	Retrieves information for a specific forecast cell
CB.GetForePercent	Returns percentiles for forecast cells
CB.Simulation	Runs a Crystal Ball simulation

CB.GetLockFitParm

This function returns a different value depending on the parameter(s) passed to the call:

- If CB.GetLockFitParm is called with only the distribution index, it returns a boolean indicating whether locking is On or Off for that distribution.
- If CB.GetLockFitParm is called with a distribution index and a parameter index, it returns the value of the locked parameter (if it is locked) or cbParmUndefined (if the parameter is not locked but can be locked or if the distribution and parameter combination is invalid for parameter locking).

Table 126 CB.GetLocktFitParm Returned Data Type

Returned Value	Returned Data Type
A boolean indicating whether locking is On (True) or Off (False) for the distribution or the actual value of the locked parameter (or cbParmUndefined if the parameter is not locked or cannot be locked)	Variant

CB.GetLockFitParm has the parameters shown in Table 127.

Table 127 CB.GetLockFitParm Parameters

Parameter	VBA Data Type	Value	Description
Dist	Integer	Distribution name constants as listed in Table 53. Valid distributions are: gamma, lognormal, Student's *t*, Weibull, binomial, and hypergeometric.	Indicates the distribution for locked parameter information
ParameterIndex (Optional)	Integer	See Table 53. Valid distribution and parameter combinations are: ● Gamma distribution — Location, Shape parameters ● Lognormal distribution — Location parameter ● Student's *t* distribution — Degrees of Freedom parameter ● Weibull distribution — Location, Shape parameters ● Binomial distribution — Trials parameter ● Hypergeometric distribution — Trials parameter	Returns the locked value setting for the specified distribution/parameter combination. If the value has been cleared or the combination is not valid, cbParmUndefined is returned.

CB.GetLockFitParm Example

The following macro statements define two variables named LockVal and useLock, lock the Location parameter of the Weibull distribution and set it to the value of -250, unlock the same parameter for the gamma distribution while setting the value of that parameter to -200, test whether the Weibull and gamma distributions have parameter locking switched on, and finally return the current values set for the Location parameters of the Weibull and gamma distributions.

```
Dim LockVal as Double
Dim UseLock as boolean

'Set the Weibull Location parameter to -250 for locking purposes and lock
'it
cb.LockFitParm cbDfaWeibull, True, 1, -250

'Set the gamma Location parameter to -200 for locking purposes but do not 'lock it
cb.LockFitParm cbDfaGamma, False, 1, -200

'Test if the Weibull distribution has parameter locking switched on;
'returns True
UseLock = CB.GetLockFitParm(cbDfaWeibull)
```

```
'Test if the gamma distribution has parameter locking switched on;
'returns False
UseLock = CB.GetLockFitParm(cbDfaGamma)

'Find the value set for the Weibull Location parameter; returns -250
LockVal=CB.GetLockFitParm(cbDfaWeibull, 1)

'Find the value set for the gamma Location parameter; returns -200
LockVal=CB.GetLockFitParm(cbDfaGamma, 1)
```

CB.GetLockFitParm Related Calls

Name	Description
CB.Fit	Fits a specific probability distribution to a selected set of data (range of cells)
CB.GetFitParm	Returns the value of the specified distribution parameter for the last CB.Fit call
CB.LockFitParm	Sets parameter locking for distributions and, optionally, specific parameters
CB.SetFitRange	Specifies the data range to be used as input for CB.Fit

CB.GetRunPrefs

This function returns run preference settings. You can call this function from either a macro or a worksheet. See Table 152 for a list of valid parameters.

Table 128 CB.GetRunPrefs Returned Data Type

Returned Value	Returned Data Type
The specified run preference setting	Variant

Note: To call this function from a worksheet, use the name CB.GetRunPrefsFN. Parameters are the ones listed in Table 152 and return the values listed there.

CB.GetRunPrefs Example 1

This example puts the current settings for the first four run preferences into cells M10 through M13.

In a macro, you can enter the line:

```
For i = 1 To 4
   Range("M9").Offset(i, 0).Value = CB.GetRunPrefs(i)
Next i
```

CB.GetRunPrefs Example 2

This example puts the current sample size into the cell.

In a worksheet cell, you can enter:

```
=CB.GetRunPrefsFN(8)
```

CB.GetRunPrefs Related Calls

Name	Description
CB.RunPrefs	Changes simulation settings using a dialog
CB.RunPrefsND	Changes simulation settings without a dialog

CB.GetScenarioAnalysisOption

This function returns specified settings made by CB.ScenarioAnalysisND or the Scenario Analysis tool.

Table 129 CB.GetScenarioAnalysisOption Returned Data Type

Returned Value	Returned Data Type
The specified Scenario Analysis settings	Variant

Table 130 CB.GetScenarioAnalysisOption Parameters

Parameter	Required?	VBA Data Type	Table
Index	Required	Integer	Table 131 on page 165
Value1	Optional	Double	Table 132 on page 166

The *Index* parameters (Table 131 on page 165) are the same as those used to make the Scenario Analysis settings.

Table 131 CB.GetScenarioAnalysisOption Index Parameter — Required, Integer

Named Constant Value	VBA Data Type	Index Value	Description
cbSceChooseTarget	Integer	1	Returns the target forecast cell to be analyzed
cbSceRangeForecastResults	Integer	2	Returns whether percentiles or forecast values are to be used for the analysis: cbSceSelectPercentiles [3] cbSceSelectForeValues [4]
cbSceSelectPercentiles	Integer	3	Works with *Value1* to return the minimum or maximum of the range of percentiles used for analysis if percentiles were used
cbSceSelectForeValues	Integer	4	Works with *Value1* to return the minimum or maximum of the range of forecast values used for analysis if forecast values were used

Named Constant Value	VBA Data Type	Index Value	Description
cbSceForeOption	Integer	5	Returns the current Scenario Analysis Options setting that indicates which forecast charts to show while running Scenario Analysis: cbSceShowDefinedFore [1], cbSceShowTargetFore [2], or cbSceHideFore [3] (Table 161 on page 200)
cbSceSimTrials	Integer	6	Returns the number of trials to run
cbSceIncludeNonTargetFore	Integer	7	When True is returned, indicates that all forecasts are included in the output table; if False, only the target forecast is included

Table 132 CB.GetScenarioAnalysisOption Value1 Parameter — Optional, Variant

Related Value	Named Constant Value	VBA Data Type	Index Value	Description
For *Index* = 3, cbSceSelectPercentiles	cbSceMinValue	Double	1	If percentiles are used, returns the minimum value of the percentile range to work on
For *Index* = 3, cbSceSelectPercentiles	cbSceMaxValue	Double	2	If percentiles are used, returns the maximum value of the percentile range to work on
For *Index* = 4, cbSceSelectForeValues	cbSceMinValue	Double	1	If forecast values are used, returns the minimum value of the value range to work on
For *Index* = 4, cbSceSelectForeValues	cbSceMaxValue	Double	2	If forecast values are used, returns the maximum value of the value range to work on

CB.GetScenarioAnalysisOption Example

These lines of example code return the options set in "CB.ScenarioAnalysisND Example" on page 201. In actual function code, the return values should be assigned to variables so they can be displayed or written to a worksheet.

```
' The following code returns the target forecast cell, B1
CB.GetScenarioAnalysisOption cbSceChooseTarget

' Returns cbSceSelectPercentiles
' Percentiles are used for analysis
CB.GetScenarioAnalysisOption cbSceRangeForecastResults

' Returns the minimum value of the percentile range (50th percentile)
CB.GetScenarioAnalysisOption cbSceSelectPercentiles, cbSceMinValue

' Returns the maximum value of the percentile range (75th percentile)
CB.GetScenarioAnalysisOption cbSceSelectPercentiles, cbSceMaxValue

' Returns the Options setting, cbSceShowTargetFore
CB.GetScenarioAnalysisOption cbSceForeOption

' Returns the number of trials to run, 500
CB.GetScenarioAnalysisOption cbSceSimTrials

' Returns whether to include only the target in the results, False
CB.GetScenarioAnalysisOption cbSceIncludeNonTargetFore
```

CB.GetScenarioAnalysisOption Related Calls

Name	Description
CB.ScenarioAnalysis	Opens the Scenario Analysis tool wizard
CB.ScenarioAnalysisND	Runs the Scenario Analysis tool without using dialogs

CB.GetTwoDSimulationOption

This function returns specified settings made by CB.TwoDSimulationND or the 2D Simulation tool.

Table 133 CB.GetTwoDSimulationOption Returned Data Type

Returned Value	Returned Data Type
The specified 2D Simulation settings	Variant

Table 134 CB.GetTwoDSimulationOption Parameters

Parameter	Required?	VBA Data Type	Table
Index	Required	Integer	Table 135
Value1	Optional	Integer	Table 136
Value2	Optional	Variant	Table 137

Table 135 CB.GetTwoDSimulationOption Index Parameter – Required, Integer

Value or Named Constant	Index	Description
cbTwoDGetTarget	1	Returns the address of the selected target forecast for the 2D simulation operation (operates on the forecast in the currently selected cell; use [cell].Select first to select the cell)
cbTwoDGetAssum	2	Works with *Value1* to enumerate the list of assumptions by type (uncertainty or variability); returns a string containg the full name (cell address) of the cells. Each successive call returns the next assumption in the list.
cbTwoDOuterSimTrials	3	Returns the number of trials in the outer (uncertainty) simulation loop
cbTwoDInnerSimTrials	4	Returns the number of trials in the inner (variability) simulation loop
cbTwoDForeOption	5	Returns the *Value1* constant that indicates the current setting for forecast chart window display options
cbTwoDOutputOption	6	Used with *Value1* to get output report options
cbTwoDPercent	7	Used with *Value1* to return current percentile settings; optionally, you can supply a Microsoft Excel range to fill with percentile values

Value or Named Constant	Index	Description
cbTwoDMSDOption	8	Returns the maximum number of uncertainty simulations to display in the output report and overlay chart

Table 136 CB.GetTwoDSimulationOption Value1 Parameter – Optional, Integer

Related Value	Value or Named Constant	Index	Description
For *Index* = 2, cbTwoDGetAssum: Integer	cbTwoDGetUncerainty	1	Used with cbTwoDGetAssum to enumerate the list of selected uncertainty assumptions
For *Index* = 2, cbTwoDGetAssum: Integer	cbTwoDGetVariability	2	Used with cbTwoDGetAssum to enumerate the list of selected variability assumptions
For Index = 3, cbTwoDOuterSim Trials	Positive whole number (integer)	n/a	Used with cbTwoDOuterSimTrials to indicate the number of trials in the outer (uncertainty) simulation
For Index = 4, cbTwoDInnerSim Trials	Positive whole number (integer)	n/a	Used with cbTwoDInnerSimTrials to indicate the number of trials in the inner (variability) simulation
For *Index* = 5, cbTwoDForeOption: Integer	cbTwoDShowDefineFore	1	Returned to indicate that all defined forecasts are displayed; the equivalent of the Show Forecasts As Defined dialog setting
For *Index* = 5, cbTwoDForeOption: Integer	cbTwoDShowTargetFore	2	Returned to indicate that only the target forecast is displayed; the equivalent of the Show Only Target Forecast dialog setting
For *Index* = 5, cbTwoDForeOption: Integer	cbTwoDHideFore	3	Returned to indicate that all forecasts are hidden; the equivalent of the Hide All Forecasts dialog setting
For Index = 6, cbTwoDOutputOption: Integer	cbTwoDShowForeStat	1	Returns a boolean *Value2* parameter to indicate whether to include forecast statistics in the output report; if followed by True, includes statistics in the report
For Index = 6, cbTwoDOutputOption: Integer	cbTwoDShowPercentiles	2	Returns a boolean *Value2* parameter to specify whether to include percentiles in the output report; if followed by True, includes percentiles in the report
For Index = 6, cbTwoDOutputOption: Integer	cbTwoDShowCapMetrics	3	Returns a boolean *Value2* parameter to specify whether to include capability metrics in the output report; if followed by True, includes capability metrics in the report
For *Index* = 7, cbTwoDPercent: Integer	cbTwoDPercentTenNinety	1	Returned by cbTwoDPercent to indicate the 10th and 90th percentiles
For *Index* = 7, cbTwoDPercent: Integer	cbTwoDPercentFiveNinetyFive	2	Returned by cbTwoDPercent to indicate the 5th and 95th percentiles
For *Index* = 7, cbTwoDPercent: Integer	cbTwoDPercentQuartiles	3	Returned by cbTwoDPercent to indicate quartiles (25%-ile increments from 0 to 100, inclusive)
For *Index* = 7, cbTwoDPercent: Integer	cbTwoDPercentQuintiles	4	Returned by cbTwoDPercent to indicate quintiles (20%-ile increments from 0 to 100, inclusive)
For *Index* = 7, cbTwoDPercent: Integer	cbTwoDPercentDeciles	5	Returned by cbTwoDPercent to indicate deciles (10%-ile increments from 0 to 100, inclusive)

Related Value	Value or Named Constant	Index	Description
For *Index* = 7, cbTwoDPercent: Integer	cbTwoDPercentIcosatiles	6	Returned by cbTwoDPercent to indicate icosatiles (5%-ile increments from 0 to 100, inclusive)
For *Index* = 7, cbTwoDPercent: Integer	cbTwoDPercentCustom	7	Returned by cbTwoDPercent to indicate custom percentiles. If cbTwoDPercentCustom is supplied as a parameter along with a valid Microsoft Excel range, fills a column-oriented range with percentiles in the custom list, starting with the cell passed in the range.
For *Index* = 8, cbTwoDMSDOption: Integer	Integer from 2 to 250, inclusive	n/a	Returned to indicate the maximum number of uncertainty simulations to display in the output report and overlay chart

Table 137 CB.GetTwoDSimulationOption Value2 Parameter – Optional, Variant

Related Value	Value or Named Constant	VBA Data Type	Index	Description
For Index = 6, cbTwoDOutputOption: Integer	True or False	Boolean	n/a	Indicates whether the specified *Value1* constant is active or not; if set to True, the specified items are included in the output report
For Value1 = cbTwoDPercentCustom	A string containing a Microsoft Excel range to hold desired percentile values	String	n/a	Returns custom percentile values specified for CB.TwoDSimulationND

CB.GetTwoDSimulationOption Example 1

This example returns a constant that indicates the current forecast display option: cbTwoDShowDefineFore, cbTwoDShowTargetFore, or cbTwoDHideFore.

```
CB.GetTwoDSimulationOption cbTwoDForeOption
```

CB.GetTwoDSimulationOption Example 2

This example returns a set of custom percentile values. The returned values are placed in the specified range, in this case, F20:H20.

```
CB.GetTwoDSimulationOption cbTwoDPercent, cbTwoDPercentCustom, "F20:H20"
```

CB.GetTwoDSimulationOption Related Calls

Name	Description
CB.TwoDSimulation	Launches the 2DSimulation tool
CB.TwoDSimulationND	Runs the 2DSimulation tool without displaying dialogs

CB.GetVersion

This function returns the version ID or build number of Crystal Ball. Optional parameters, *VersionType* and *Index*, determine what is returned and how much of it appears.

Table 138 CB.GetVersion Returned Data Type

Returned Value	Returned Data Type
The version ID number or build number of Crystal Ball as a string.	Variant

Table 139 CB.GetVersion Parameters

Parameter	VBA Data Type	Value or Named Constant	Index	Description
VersionType (Optional)	Variant	A constant as specified in the following two rows; if 0 or unspecified, the Crystal Ball version ID is returned as a string (equivalent to cbProductVersion)	n/a	n/a
VersionType (Optional)	Variant	cbProductVersion	0	Returns all or part of the Crystal Ball version ID, for example, 11.1.1.1.00; can be used with Index to return only a portion of the version ID
VersionType (Optional)	Variant	cbFullBuildVersion	1	Returns all or part of the Crystal Ball build number, for example, 11.1.159.0; can be used with Index to return only a portion of the build number
Index (Optional)	Integer	An integer from 1 to the number of sections in the string returned by VersionType.	n/a	Returns the specified portion of the version ID or build number, counting from left to right. If Index = 3, 159 would be returned for cbFullBuildVersion in the previous example; the maximum value for Index in this example would be 4. If Index = 0 or is not specified, the full string for VersionType is returned.

CB.GetVersion Example

For Crystal Ball version 11.1.1.0.00, this example places the string "11.1.1.0.00" in cell A2 and a string with the full build number, 11.1.63.0, into cell B2.

```
Range("A2").Value = CB.GetVersion
Range("B2").Value = CB.GetVersion(cbFullBuildVersion)
```

CB.GetVersion Related Calls

Name	Description
CB.GetWorksheetVersion	Returns the Crystal Ball data version of the active or specified worksheet

CB.GetWorksheetVersion

This function returns the Crystal Ball data version ID or build number for the active or specified worksheet. It has three optional parameters. The first optional parameter, *Worksheet*, indicates the target worksheet. If omitted, information is returned for the active worksheet. The second optional parameter, *VersionType*, and the third optional parameter, *Index*, can return the full or partial version ID or build number as a string.

Note: For Crystal Ball versions 7.0 through 7.1.2, the version ID returned is "7.0.0". This string changed starting with data created by Crystal Ball version 7.2.

Table 140 CB.GetWorksheetVersion Returned Data Type

Returned Value	Returned Data Type
The Crystal Ball data version ID or build number returned for the active or specified worksheet	Variant

Table 141 CB.GetWorksheetVersion Parameters

Parameter	VBA Data Type	Value	Description
Worksheet (Optional)	Worksheet	A Microsoft Excel object indicating the target workbook and worksheet	Indicates the target worksheet whose version to return. If omitted, returns the version of the currently active worksheet.
VersionType (Optional)	Variant	0 or 1	Returns the Crystal Ball version ID or build number as a string. For details, see Table 139.
Index (Optional)	Integer	An integer from 1 to the number of sections in the string returned by *VersionType*	Returns the specified portion of the version ID or build number, counting from left to right. For details, see Table 139.

CB.GetWorksheetVersion Example

This example puts the third section of the build number of the version of Crystal Ball used to create the data on Sheet1 in cell A2 of the active worksheet and puts the version ID string of the version of Crystal Ball used to create Sheet 2 in cell B2. If the full build number is 11.1.159.0 and the version ID number is 11.1.1.1.00, 159 is placed in cell A2 and "11.1.1.1.00" is placed in cell B2.

```
Range("A2").Value = CB.GetWorksheetVersion(Sheet1, cbFullBuildVersion, 3)
Range("B2").Value = CB.GetWorksheetVersion(Sheet2, cbProductVersion)
```

CB.GetWorksheetVersion Related Calls

Name	Description
CB.GetVersion	Returns all or part of the current Crystal Ball version as a full build number or version ID number

CB.IsCBObject

This function determines if a worksheet or workbook contains Crystal Ball data.

Table 142 CB.IsCBObject Returned Data Type

Returned Value	Returned Data Type
Whether the current worksheet or workbook contains Crystal Ball data	Boolean

Most of the Developer Kit subroutines and function will not work if the active worksheet or workbook does not contain Crystal Ball data.

Table 143 CB.IsCBObject Parameters

Parameter	VBA Data Type	Value	Description
AnObject	Variant	A Microsoft Excel workbook or worksheet. For example: Worksheet("Sheet1")	Points to the worksheet or workbook to check for Crystal Ball data

CB.IsCBObject Example

This example checks to see whether user_sheet is a worksheet object (and not a module sheet or something else). If it is a worksheet, it checks the worksheet and its parent (the workbook) for Crystal Ball data. If user_sheet is not a worksheet or if no Crystal Ball data exist on either the worksheet or the workbook, it displays an error message and exits the macro. Otherwise, it continues.

```
Set user_sheet = Worksheets("Sheet2")
If (Not (TypeName(user_sheet) = "Worksheet")) Or _
   (Not (CB.IsCBObject(user_sheet) Or _
   CB.IsCBObject(user_sheet.Parent))) Then
   MsgBox "Active document does not contain Crystal Ball Data."
   Exit Sub
End If
```

CB.IsCBObject Related Calls

Name	Description
CB.MacroResult	Tells you if a Crystal Ball macro call worked correctly

CB.Iterations

This function returns the number of trials performed thus far in the simulation. You can call this function from either a macro or a worksheet.

Table 144 CB.Iterations Returned Data Type

Returned Value	Returned Data Type
The number of trials already run in a simulation	Variant

Note: To call this function from a worksheet, use the name CB.IterationsFN.

CB.Iterations Example 1

This example runs a simulation for 1000 trials, 100 trials at a time. After each 100 trials, the current number of trials and the mean of the forecast in B7 at that point are put into cells on sheet 4.

```
Worksheets("sheet1").Activate
For i = 1 To 10
   CB.Simulation 100
   Worksheets("sheet4").Range("A1").Offset(i).Value = CB.Iterations
       Worksheets("sheet4").Range("B1").Offset(i).Value=_
      CB.GetForeStat(Range("B7"), 2)
Next i
```

CB.Iterations Example 2

In a worksheet cell, you can enter:

```
=CB.IterationsFN()
```

Example 2 enters the number of trials run so far into the cell.

CB.Iterations Related Call

Name	Description
CB.Simulation	Runs a simulation for a specified number of iterations

CB.LockFitParm

This subroutine enables you to enter specific parameters for selected distributions when selecting distributions for fitting. This can be helpful when locking to a specific value for location, shape, degrees of freedom or other parameters can improve accuracy.

Table 145 CB.LockFitParm Parameters with Values

Parameter	VBA Data Type	Value	Description
Dist	Integer	See Table 53 for constant equivalents to the distributions listed in the following table cell (under Description)	Specifies the distribution with parameters to lock. The following distributions from Table 53 can be used: ● Gamma distribution — Location, Shape parameters ● Lognormal distribution — Location parameter ● Student's t distribution — Degrees of Freedom parameter ● Weibull distribution — Location, Shape parameters ● Binomial distribution — Trials parameter ● Hypergeometric distribution — Trials parameter
Use	Boolean	True or False	Indicates if the lock value should be used or not; if set to True, the lock value is used
ParameterIndex (Optional)	Integer	See Table 53 to determine index values for target parameters	The index of the distribution parameter to lock, taken from Table 53
Value	Double	A positive real number	The actual value to be used for the specified parameter. Cell references cannot be used here. To clear the locked value, pass the constant cbParmUndefined.

Locked values are stored as if you had set them using Crystal Ball dialogs. They do not need to be called before every CB.Fit call.

CB.LockFitParm Example

The following macro statements define a variable named Result, lock the Location parameter of the gamma distribution, clear any previous values from the Shape parameter of the gamma distribution, run the fit and return results. Notice that CB.SetFitRange is used before CBLockFitParm:

```
Dim Result as Double

'Define a range of data for fitting
CB.SetFitRange ("A1:A30")

'Lock the Location of Gamma to 0
CB.LockFitParm cbDfaGamma, True, 1, 0

'Clear any locked values for the Shape of Gamma
CB.LockFitParm cbDfaGamma, True, 3, cbParmUndefined

'Run the fit and return the result
Result = CB.Fit(cbDfaGamma, cbFitAndersonDarling, False)
```

CB.LockFitParm Related Calls

Name	Description
CB.Fit	Fits a specific probability distribution to a selected set of data (range of cells)
CB.GetFitParm	Returns the value of the specified distribution parameter for the last CB.Fit call
CB.GetLockFitParm	Returns whether a distribution and, optionally, specific parameters have locked values for fitting purposes
CB.SetFitRange	Specifies the data range to be used as input for CB.Fit

CB.MacroResult

This function returns information about whether other Crystal Ball macro calls worked correctly. Most of the calls described in this document set a result value that CB.MacroResult returns. CB.MacroResult does not clear the result value until another Crystal Ball call changes the value.

Table 146 CB.MacroResult Returned Data Type

Returned Value	Returned Data Type
An error code that indicates what type of error the previous macro call encountered	Variant

For a list of the named constants and values that CB.MacroResult returns, see Table 149. In VBA subroutines and functions, you can check the returned value against either the named constants or values. All returned values are negative integers (or 0 if no errors occur).

Note: This function is for compatibility with the 2000.x (5.x) versions of the Crystal Ball Developer Kit. CB.MacroResultDetail is recommended instead because it provides a wider range of errors.

CB.MacroResult Example

This example subroutine, checkresult, checks the last Crystal Ball macro call. It uses the CB.MacroResult return value to check the status of the last call and display the appropriate message.

```
Sub checkresult
   Select case CB.MacroResult
   case cbErrNone
      MsgBox "Completed OK"
   case cbErrBadValue
      MsgBox "Bad parameter value"
   case cbErrNoForecastInCell
      MsgBox "No forecast in active cell"
   case cbErrNotReady
      MsgBox "No simulation yet"
   case cbErrNoAssumptionInCell
```

```
      MsgBox "Missing assumptions in selected cell(s)"
   case cbErrBadSelector
      MsgBox "Bad value in parameter"
   case cbErrBadCommand
      MsgBox "Bad value in parameter parm_number"
   case cbErrNoCBSheets
      MsgBox "No worksheets with CB information to save"
   case cbErrTooManyAssumptions
      MsgBox "Too many assumptions"
   case cbErrTooManyForecasts
      MsgBox "Too many forecasts"
   case cbErrBadCorrelation
      MsgBox "Invalid correlation(s)"
   case cbErrBadAssumption
      MsgBox "Invalid assumption parameter(s)"
   case cbErrCallingDLL
      MsgBox "Could not find the DLL"
   case cbErrUnexpected
      MsgBox "An unexpected error occurred"
   End Select
End Sub
```

CB.MacroResult Related Call

Name	Description
CB.MacroResultDetail	Indicates if a macro call worked correctly, including detailed error information

CB.MacroResultDetail

This function returns detailed information about whether other Crystal Ball macro calls worked correctly and can include additional information for compatibility and diagnostic purposes.

Table 147 CB.MacroResultDetail Returned Data Type

Returned Value	Returned Data Type
The value of the specified property of CBMacroResultDetail as indicated in Table 148 on page 176	Variant

CB.MacroResultDetail is different from other Developer Kit subroutines and functions because its parameters are properties of a type class instead of constants. See the example for appropriate syntax.

Table 148 CB.MacroResultDetail Properties

Property	VBA Data Type	Description
OldValue	Integer	The value returned by CB.MacroResult, listed in Table 149
NewValue	Integer	The value returned by CB.MacroResultDetail.NewValue in this version of Crystal Ball

Property	VBA Data Type	Description
Msg	String	A string that summarizes the error
MsgDetails	String	A message with more information about the error
MsgTitle	String	The command for which these results are returned
DialogResult	String	Non-localized strings describing the actions in a dialog: "Yes," "No," "OK," "Cancel," and "None. " "None" indicates that no dialog appeared
ElapsedTimeInMS	Long	The time in milliseconds required to process the command

To see a list of the named constants and values returned by CB.MacroResultDetail.NewValue, open Crystal Ball online help and find Crystal Ball Error Messages in the Contents. In VBA subroutines and functions, you can check the returned value against either the named constants or values.

Table 149 Values Returned by CB.MacroResult or CB.MacroResultDetail.OldValue

Returned Constant	Returned Value	Description
cbErrNone	0	The previous Crystal Ball command worked perfectly
cbErrBadValue	-1	A parameter had the wrong value
cbErrNoForecastInCell	-2	The command required a forecast cell to be active; the active cell did not contain a forecast
cbErrNotReady	-4	Crystal Ball has not yet run a simulation so the requested macro cannot be run
cbErrNoAssumptionInCell	-5	The command required one or two assumption cells to be selected. One or more selected cells did not contain assumptions.
cbErrBadSelector	-6	A parameter that was restricted to a small number of integers had a bad value
cbErrBadCommand	-7	The parm_number parameter had a bad value
cbErrUnsavedSheet	-8	The operation required all worksheets to be saved but one or more sheets were not saved
cbErrNoCBSheets	-9	There were no worksheets with Crystal Ball information to save
cbErrRestoreResult	-10	Restore Results failed
cbErrTooManyAssumptions	-11	Crystal Ball can store no more assumptions
cbErrTooManyForecasts	-12	Crystal Ball can store no more forecasts
cbErrBadCorrelation	-13	One or more correlations were invalid
cbErrBadAssumption	-14	One or more assumption parameters were invalid
cbErrUnexpected	-101	An unexpected error occurred, such as a formula cell was used for a macro that required a single-value cell

CB.MacroResultDetail Example

This subroutine, checkresult, clears the specified range, defines a normal distribution in cell A1, defines a forecast in cell A2, runs a simulation and displays a dialog for each of the CB.MacroResultDetail properties. Then, it clears the Crystal Ball data in cells A1 and A2 and stops.

```
Sub checkresult()

Range("A1:A2").Select
CB.ResetND
CB.ClearDataND
Range("A1:A2").Clear

Range("A1").Select
ActiveCell.Value = 0
CB.DefineAssumND cbDfaNormal, 0, 1

Range("A2").Select
ActiveCell.Value = "=A1"
CB.DefineForeND

CB.Simulation 1000
MsgBox (CB.MacroResultDetail.DialogResult)
MsgBox (CB.MacroResultDetail.ElapsedTimeInMS)
MsgBox (CB.MacroResultDetail.Msg)
MsgBox (CB.MacroResultDetail.MsgDetails)
MsgBox (CB.MacroResultDetail.MsgTitle)
MsgBox (CB.MacroResultDetail.NewValue)
MsgBox (CB.MacroResultDetail.OldValue)

Range("A1:A2").Select
CB.ResetND
CB.ClearDataND
Range("A1:A2").Clear

End Sub
```

CB.MacroResultDetail Related Call

Name	Description
CB.MacroResult	Indicates if a macro call worked correctly, including limited error information

CB.OpenChart

This call opens a chart of the specified type and chart ID. The chart ID is optional. If no charts of the specified type exist, one is created and is selected. If the chart ID is not specified, the currently selected chart is opened. If the specified chart is not found, a chooser dialog appears. No error is returned.

Currently, this call only works on current and not restored results.

Table 150 CB.OpenChart Parameters with Values

Parameter	VBA Data Type	Value	Description
SimChartType	Integer	cbChtOverlay = 1, cbChtScatter = 2, cbChtSensitiv = 3, cbChtTrend = 4	Specifies the type of chart you are opening — overlay, scatter, sensitivity, or trend — using the named constant and index values listed in the Value column
ChartID (Optional)	String	"*Chart ID*"	The chart ID returned by CB.EnumChart (Table 60) entered directly or using CB.EnumChart

CB.OpenChart Example 1

This example opens the specified overlay chart, trend chart, and sensitivity chart.

```
CB.OpenChart cbChtOverlay, "Overlay Chart 1"
CB.OpenChart cbChtTrend, "Trend Chart 1"
CB.OpenChart cbChtSensitiv, "Sensitivity Chart 1"
```

CB.OpenChart Example 2

This example runs a simulation, calls CB.CheckData (because a CB.Enum... call follows), and then opens the next overlay chart in the workbook.

```
CB.Simulation 1000
CB.CheckData
CB.OpenChart cbChtOverlay, CB.EnumChart(cbChtOverlay)
```

CB.OpenChart Related Calls

Name	Description
CB.CloseAllCharts	Closes all open chart windows; equivalent to Analyze, then Close All
CB.CloseChart	Closes the selected chart of the given type
CB.EnumChart	Returns the unique name of the next chart of the specified type
CB.SelectChart	Selects the chart with the specified chart ID

CB.OpenFore

This subroutine opens the forecast window for the selected cell.

The worksheet must be active, a simulation run, and a forecast cell selected for Crystal Ball to run this command.

CB.OpenFore Example

This example runs a simulation of 1000 trials, activates the Model worksheet of the Futura with OptQuest.xls workbook, and selects cell C9, a forecast. It then opens the forecast window associated with that cell.

```
CB.Simulation 1000
Workbooks("Futura with OptQuest.xls").Worksheets("Model").Activate
Range("C9").Select
CB.OpenFore
```

CB.OpenFore Related Calls

Name	Description
CB.CloseFore	Closes the forecast window for the selected cell
CB.Simulation	Runs a Crystal Ball simulation

CB.OpenSelection

This subroutine is the equivalent of Analyze, then Open Selected Cells. It opens all charts defined for cells in the selected range.

Before calling this subroutine, run a simulation.

CB.OpenSelection Example

This example runs a simulation of 5000 trials and then opens whatever charts are defined for cells C9, C10, C11, and C12.

```
CB.Simulation 5000
Range("C9:C12").Select
CB.OpenSelection
```

CB.OpenSelection Related Calls

Name	Description
CB.OpenFore	Opens a forecast chart for the selected cell
CB.Simulation	Runs a Crystal Ball simulation

CB.OpenSensitiv

This call opens the selected sensitivity chart. If no sensitivity charts have been created for a model, one is created for the first forecast and opened. If more than one chart is found but one is not selected, a chooser dialog is displayed.

Note: This call is included for compatibility with Crystal Ball 2000.x (5.x). However, it differs somewhat from earlier versions. For details, see "CB.OpenSensitiv" on page 247). To open a chart of a specific type by its chart ID, use CB.OpenChart ("CB.OpenChart" on page 178).

CB.OpenSensitiv Example

This example runs a simulation, calls CB.CheckData (because a CB.Enum... call follows), selects the next sensitivity chart in the workbook and then opens it.

```
CB.Simulation 1000
CB.CheckData
CB.SelectChart cbChtSensitiv, CB.EnumChart(cbChtSensitiv)
CB.OpenSensitiv
```

CB.OpenSensitiv Related Calls

Name	Description
CB.CloseSensitiv	Closes the selected sensitivity chart
CB.CopySensitiv	Copies the selected sensitivity chart to the clipboard
CB.EnumChart	Returns the unique name of the next chart of the specified type
CB.SelectChart	Selects the chart with the specified chart ID

CB.OpenTrend

This call opens the selected trend chart. If no trend charts have been created for a model, one is created. It is blank until one or more forecasts are added with the appropriate CB.TrendPrefsND call. If more than one chart is found but one is not selected, a chooser dialog appears.

Note: This call is included for compatibility with Crystal Ball 2000.x (5.x). However, it differs somewhat from earlier versions. For details, see "CB.OpenTrend" on page 247). To open a chart of a specific type by its chart ID, use CB.OpenChart ("CB.OpenChart" on page 178).

CB.OpenTrend Example

This example runs a simulation, calls CB.CheckData (because a CB.Enum... call follows), selects the next trend chart in the workbook and then opens it.

```
CB.Simulation 1000
CB.CheckData
CB.SelectChart cbChtTrend, CB.EnumChart(cbChtTrend)
CB.OpenTrend
```

CB.OpenTrend Related Calls

Name	Description
CB.CloseTrend	Closes the selected trend chart
CB.EnumChart	Returns the unique name of the next chart of the specified type
CB.SelectChart	Selects the chart with the specified chart ID

CB.PasteData

This subroutine pastes assumption, decision, or forecast data from ranges specified with CB.CopyData and CB.CopyDataND. This subroutine, together with CB.CopyData and CB.CopyDataND, helps you to quickly set up rows or columns of assumptions with similar distribution types or decision variables with similar bounds.

The worksheet must be active for Crystal Ball to run this command. Assumption and decision variable data can be pasted into simple value cells. or blank cells. Forecasts may be pasted only into value or formula cells.

If assumptions or decision variables are pasted into a range that is totally blank, any underlying Microsoft Excel values are pasted along with the Crystal Ball data.

If the range selected for pasting contains more value or formula cells than are selected for copying, the cells will repeat. For example, if the CopyData range includes two assumptions and the PasteData range includes three value cells, the first and second assumptions are pasted and then the first assumption is repeated.

Note: Before calling this subroutine, copy Crystal Ball data using CB.CopyData or CB.CopyDataND.

CB.PasteData Example 1

This example copies the Crystal Ball data from cell C1 and pastes it into cell D1.

```
Range("C1").Select
CB.CopyData
```

```
Range("D1").Select
CB.PasteData
```

CB.PasteData Example 2

This example copies the assumption data from cells B5, B6, and B7, and then pastes it into the corresponding cells D1, D2, and D3 (provided those cells contain values and not formulas).

```
'In the following command, B5, B6, and B7 are all assumptions
Range("B5:B7").Select

CB.CopyData
Range("D1:D3").Select
CB.PasteData
```

CB.PasteData Example 3

This example copies the Crystal Ball data from cell B5 and pastes it three times: into cells D1, D2, and D3.

```
Range("B5").Select
CB.CopyData
Range("D1:D3").Select
CB.PasteData
```

CB.PasteData Related Calls

Name	Description
CB.CopyData	Copies Crystal Ball data in specified range

CB.Reset

This subroutine resets the current simulation, prompting you to confirm the reset. When you confirm, it closes all forecast windows, the trend chart, the overlay chart, and the sensitivity chart; then it clears results from the previous simulation.

This is very useful to use before calls to subroutines and functions that require that a simulation be reset, such as CB.Freeze, although you might prefer CB.ResetND because it resets automatically without user response. You can also call this subroutine to make sure that simulations start at Trial 1.

You do not have to run a simulation before you call this subroutine. However, it is convenient to run to ensure that no prior simulation results are around.

Note: If you run a simulation using the CB.Simulation subroutine without resetting, Crystal Ball runs additional trials and adds the results to the existing results. This is different from the user interface command, which tells you it has reached the maximum number of trials and requires you to reset the simulation.

CB.Reset Example

This example prompts you to confirm and then clears any simulation data and closes all forecast windows and charts.

```
CB.Reset
```

CB.Reset Related Calls

Name	Description
CB.ResetND	Resets a simulation without confirmation
CB.Simulation	Runs a simulation for specified number of iterations

CB.ResetND

This subroutine resets the current simulation without prompting you to confirm the reset. It automatically closes all forecast windows, the trend chart, the overlay chart, and the sensitivity chart; then it clears results from the previous simulation.

This is very useful to use before calls to subroutines and functions that require that a simulation be reset, such as CB.Freeze. You can also call this subroutine to make sure that simulations start at Trial 1. For users to confirm the reset, use CB.Reset because it includes a prompt before it runs.

You do not have to run a simulation before you call this subroutine. However, it is convenient to run to ensure that no prior simulation results are around.

Note: If you run a simulation using the CB.Simulation subroutine without resetting, Crystal Ballruns additional trials and adds the results to the existing results. This is different from the user interface command, which tells you it has reached the maximum number of trials and requires you to reset the simulation.

CB.ResetND Example

This example clears any simulation data and closes all forecast windows and charts without prompting to confirm the closures.

```
CB.ResetND
```

CB.ResetND Related Calls

Name	Description
CB.Reset	Resets a simulation after prompting
CB.Simulation	Runs a simulation for specified number of iterations

CB.RestoreResults

This subroutine opens the Restore Results dialog so you can restore previously saved simulation results (any results data and charts created — but not necessarily open — at the time the results are saved).

You can restore any number of results files at one time, but they must be opened one at a time by calling them with CB.RestoreResults. You do not need to reset the current simulation before you run this subroutine. You can restore results while a simulation is running, completed, or stopped.

To run this subroutine, you must have a saved .cbr file, or CB.MacroResult returns cbErrRestoreResult.

CB.RestoreResults is similar to CB.RestoreRun, available in Crystal Ball 2000.5 (5.5). However, CB.RestoreRun and CB.RestoreRunND restored an entire simulation (saved with CB.SaveRun or CB.SaveRunND), not just the results. The existing simulation had to be reset before the saved simulation could be loaded and only one simulation could be restored at a time.

See the *Oracle Crystal Ball User's Guide* for more information on saving simulation results.

CB.RestoreResults Example

This example automatically opens the Restore Results dialog to restore previously saved simulation results.

```
CB.RestoreResults
```

CB.RestoreResults Related Calls

Name	Description
CB.RestoreResultsND	Restores simulation results from a file without a dialog
CB.SaveResults	Saves current simulation results data, charts and reports using a dialog
CB.SaveResultsND	Saves current simulation results data, charts, and reports without a dialog

CB.RestoreResultsND

This subroutine automatically restores simulation results from the file specified by *FileName*. See CB.RestoreResults for more information about what is restored.

To run this subroutine, you must have a saved .cbr file, or CB.MacroResult returns cbErrRestoreResult.

You can restore any number of results files at one time (but they must be restored one at a time). You do not need to reset the current simulation before you run this subroutine. You can restore results while a simulation is running, completed, or stopped.

If the file could not be read or is not a Crystal Ball results file, CB.MacroResult returns cbErrRestoreResult.

CB.RestoreResultsND is similar to CB.RestoreRunND, available in Crystal Ball 2000.5 (5.5). However, CB.RestoreRun and CB.RestoreRunND restored an entire simulation (saved with CB.SaveRun or CB.SaveRunND), not just the results. The existing simulation had to be reset before the saved simulation could be loaded and only one simulation could be restored at a time.

Table 151 CB.RestoreResultsND Parameters

Parameter	VBA Data Type	Value	Description
FileName	String	A path surrounded by quotes (""), such as "C:\temp\sheet2.cbr"	Points to the saved results file to restore

See the *Oracle Crystal Ball User's Guide* for more information on saving simulation results.

CB.RestoreResultsND Example

This example automatically restores the previously saved simulation file, saveme.cbr.

```
CB.RestoreResultsND "C:\temp\saveme.cbr"
```

CB.RestoreResultsND Related Calls

Name	Description
CB.RestoreResults	Restores simulation results from a file using a dialog
CB.SaveResults	Saves current simulation results using a dialog
CB.SaveResultsND	Saves current simulation results without a dialog

CB.RunPrefs

This subroutine opens the Run Preferences dialog so you can change simulation settings.

CB.RunPrefs Example

This example opens the Run Preferences dialog.

```
CB.RunPrefs
```

CB.RunPrefs Related Calls

Name	Description
CB.GetRunPrefs	Returns a run preference setting
CB.RunPrefsND	Changes run preferences settings without a dialog

CB.RunPrefsND

This subroutine sets preferences corresponding to the Run Preferences dialog. Any run preferences not explicitly defined default to the current setting in the Run Preferences dialog.

See the *Oracle Crystal Ball User's Guide* for more information on setting run preferences.

Table 152 CB.RunPrefsND Parameters

Parameter	VBA Data Type	Value	Description
Index	Integer	See Table 153	Works with the *Value1* and *Value2* parameters to set Crystal Ball run preferences
Value1	Variant	See Table 154	Works with the *Index* parameter to set Crystal Ball run preferences
Value2 (Optional)	Variant	Any positive whole number	For *Index* = 4, cbRunSameRandoms: Long — Sets the initial seed value for the random number generator

The following *Index* named constant and index values (Table 153) work with the *Value1* and *Value2* parameters to set Crystal Ball run preferences.

Table 153 CB.RunPrefsND Index Parameter Values — Required, Integer

Named Constant Value	Index Value	Description
cbRunMaxTrials	1	Sets the maximum number of trials to the number in *Value1*
cbRunStopOnError	2	Sets whether the simulation stops on calculation errors in forecast cells, according to *Value1*
cbRunSameRandoms	4	Sets whether the random number generator uses the same sequence of random numbers (*Value1*) and sets the initial seed value (*Value2*)
cbRunSamplingMethod	7	Selects whether to use the Latin Hypercube or Monte Carlo sampling method, according to *Value1*

Named Constant Value	Index Value	Description
cbRunSampleSize	8	Sets the Latin Hypercube sampling method's sample size to *Value1*
cbRunCorrelationsOff	10	Sets whether to turn off correlations, according to *Value1*
cbRunMinimizeHost	26	Sets whether to minimize Microsoft Excel during simulations, according to *Value1*. **Note:** This constant only applies in Normal and Demo speeds; it does not apply in Extreme speed.
cbRunMinimizeSheets	27	Sets whether to minimize worksheets during simulations, according to *Value1*. **Note:** This constant only applies in Normal and Demo speeds; it does not apply in Extreme speed.
cbRunHideForecasts	28	Sets whether to suppress forecast and assumption windows during simulations (as well as overlay, trend, and sensitivity charts), according to *Value1*
cbRunMode	29	Sets whether to run in Extreme, Normal, or Demo speed, according to *Value1*; note that Extreme speed requires an Crystal Ball Decision Optimizer license
cbRunRandomSeed	31	Specifies the initial random seed value, according to *Value1*
cbRunPrecisionControl	40	Sets whether to stop a simulation when the forecast precision reaches a set confidence level; only takes effect after a reset
cbRunPrecisionConfidence	46	Sets the confidence level for precision control to *Value1*
cbRunReversePercentiles	47	Changes the interpretation of percentiles, according to *Value1*
cbRunFormatPercentiles	48	Indicates how to format percentiles, according to *Value1*
cbRunSaveAssumptionValues	49	Indicates whether to store assumption values for later analysis (such as sensitivity analysis) according to *Value1*
cbRunLeaveOpenOnReset	50	Sets whether to leave the Crystal Ball Control Panel open when a simulation is reset, according to *Value1*
cbRunUserMacros	51	Specifies whether or not to run user-defined macros, according to *Value1*
cbRunCapMetrics	52	Sets whether the UI should display capability metric related information, according to *Value1*
cbRunCapMetricsShort	53	Determines if the underlying forecast data should be considered short or long term, according to *Value1*
cbRunCapMetricsShift	54	Sets the shift value used for calculating Zst and Zlt, according to *Value1*
cbRunCapMetricsCalcType	55	Determines if the metrics should be calculated from the fitted curve or the data, according to *Value1*
cbRunCapMetricsNormalityThresh	56	Specifies the p-value threshold used for normality testing, according to *Value1*
cbRunCapMetricsNormalCalcOptions	57	Specifies whether to calculate the data from the fitted curve or the data if the forecast values are not normal, according to *Value1*; only valid if cbRunCapMetricsCalcType = 0

Named Constant Value	Index Value	Description
cbRunAlertLevel	62	Specifies which Crystal Ball alert messages to show according to *Value1*; optionally, clears all Do Not Show Again preferences to default values if *Value2* = True (*Value2* default = False)

Note: The following *Index* constants used in Crystal Ball 2000.5 (5.5) are now obsolete: cbRunBurstMode = 5, cbRunResetToOriginal = 6, cbRunSensitivAnalysis = 9, cbRunRetainUnsortedTrials = 11, cbRunMacroBeforeSimul = 21, cbRunMacroBeforeRecalc = 22, cbRunMacroAfterRecalc = 23, cbRunMacroAfterSimul = 24, cbRunParadiseServer = 30, cbRunBurstAmount = 33, cbRunPrecisionWithin = 45. cbRunPrecisionMean = 41, used in versions before Crystal Ball 2000.5 (5.5), is now obsolete. In Crystal Ball 2000.5 (5.5), values for cbRunMode [29] were boolean; in later versions of Crystal Ball, the values are Integer. Models from Crystal Ball 2000.5 (5.5) that use this constant will not run successfully in later versions of Crystal Ball until the constant values are edited.

The following *Value1* values (Table 154) are used with the *Index* parameter to set Crystal Ball run preferences.

Table 154 CB.RunPrefsND Value1 Parameter Values — Required, Variant

Used With Specified Values of Index	Value	Description
For *Index* = 1, cbRunMaxTrials: Long	Any positive whole number	Sets the maximum number of trials
For *Index* = 2, cbRunStopOnError: Boolean	True or False	True stops a simulation on calculation errors in forecast cells. False does not.
For *Index* = 4, cbRunSameRandoms: Boolean	True or False	True uses the same sequence of random numbers. False does not.
For *Index* = 7, cbRunSamplingMethod: Boolean	True or False OR cbSamLatinCube = True; or cbSamMonteCarlo = False	True uses Latin Hypercube. False uses Monte Carlo.
For *Index* = 8, cbRunSampleSize: Integer	A whole number between 10 and 5000	Sets the sample size for Latin Hypercube sampling
For *Index* = 10, cbRunCorrelationsOff: Boolean	True or False	True turns off correlations. False turns on correlations.
For *Index* = 26, cbRunMinimizeHost: Boolean	True or False	True minimizes Microsoft Excel during simulations. False does not.
For *Index* = 27, cbRunMinimizeSheets: Boolean	True or False	False minimizes worksheets during simulations. True does not.
For *Index* = 28, cbRunHideForecasts: Boolean	True or False	True suppresses forecast and all other chart windows during simulations. False does not.

Used With Specified Values of Index	Value	Description
For *Index* = 29, cbRunMode: Integer	cbRunExtremeSpeed = 1, cbRunNormalSpeed = 2, cbRunDemoSpeed = 3	Sets Crystal Ball to run in Extreme speed (if available), Normal speed, or Demo speed depending on the named constant or index value listed in the Value column
For *Index* = 31, cbRunRandomSeed: Long	Any positive whole number	Sets the seed value for using the same random sequence of numbers
For *Index* = 40, cbRunPrecisionControl: Boolean	True or False	True uses precision control to stop the simulation. False uses the maximum number of trials.
For *Index* = 46, cbRunPrecision Confidence: Double	Any value between 0.01 and 99. 99	Setting for cbRunPrecisionConfidence; sets the confidence level (as a percentage) for precision control
For *Index* = 47, cbRunReversePercentiles: Boolean	True or False	True makes percentiles represent the percentage of the distribution greater than the percentile value. False makes percentiles represent the percentage less than the value.
For *Index* = 48, cbRunFormatPercentiles: Boolean	True or False	Determines whether percentiles are indicated by a percent sign (10%) or by a P before the value (P10); True uses the P and False uses the % sign.
For *Index* = 49, cbRunSaveAssumptionValues: Boolean	True or False	True saves assumption values for later analysis (such as sensitivity analysis). False discards them.
For *Index* = 50, cbRunLeaveOpenOnReset: Boolean	True or False	True leaves the CB Control Panel open when a simulation is reset. False closes it.
For *Index* = 51, cbRunUserMacros: Boolean	True or False	Indicates whether or not to run user-defined macros; True runs them and False does not run them.
For *Index* = 52, cbRunCapMetrics: Boolean	True or False	Sets whether the UI should display capability metric related information; True shows metrics and False does not show capability metrics.
For *Index* = 53, cbRunCapMetricsShort: Boolean	True or False	Determines if the underlying forecast data should be considered short- or long-term; True = short-term data and False = long-term data.
For *Index* = 54, cbRunCapMetricsShift	A numeric value indicating short-term/long-term shift	Sets the shift value used for calculating Zst and Zlt; the default shift used in the UI is 1.5
For *Index* = 55, cbRunCapMetricsCalcType: Integer	0 or 1	Determines if the capability metrics should be calculated from the fitted curve or the data; 0 = calculate from fitted curve and 1 = calculate from data
For *Index* = 56, cbRunCapMetricsNormalityThresh: Double	Any positive value between 0 and 1	Specifies the p-value threshold used for normality testing
For *Index* = 57, cbRunCapMetricsNormalCalcOptions: Integer	0 or 1	Determines if the capability metrics should be calculated from the fitted curve or the data if the distribution is not normal; 0 = calculate from fitted curve and 1 = calculate from data

Used With Specified Values of Index	Value	Description
For *Index* = 62, cbRunAlertLevel: Integer	cbRunAlertShowAll = 0, cbRunAlertShowImportant = 1, cbRunAlertShowNone = 2	Sets Crystal Ball to show alert messages according to the named constant or index value listed in the Value column

CB.RunPrefsND Example 1

This example sets the simulation to use the same sequence of random numbers (with an initial seed value of 888) and use Monte Carlo simulation. It then runs the simulation of 100 trials.

```
CB.RunPrefsND cbRunSameRandoms, True, 888
CB.RunPrefsND cbRunSamplingMethod, cbSamMonteCarlo
CB.Simulation 100
```

CB.RunPrefsND Example 2

This example shows how to use *Index* = 62, cbRunAlertLevel.

```
' Set Crystal Ball to show only important alerts
CB.RunPrefsND cbRunAlertLevel, cbRunAlertShowImportant

' Set Crystal Ball to show all alerts, and reset all
'"Do not show again" alert dialog checkboxes
CB.RunPrefsND cbRunAlertLevel, cbRunAlertShowAll, True
```

CB.RunPrefsND Related Calls

Name	Description
CB.GetRunPrefs	Returns a run preference setting
CB.RunPrefs	Changes run preference settings using a dialog

CB.RuntimeMode

This subroutine turns off the Crystal Ball toolbar and menus so you can create applications with your own custom interface for novice Crystal Ball users. These custom applications can be written with the Crystal Ball Developer Kit to run on Crystal Ball version 7.2.2 or later, including 11.*x* or later.

You can give users the ability to run, reset, and single-step through simulations by creating your own user interface controls with Visual Basic or VBA that call appropriate Crystal Ball Developer Kit subroutines and functions. You can also use the Crystal Ball Developer Kit macros to create controls that extract data, create charts and reports, and more. When you run simulations, the Crystal Ball Control Panel is switched off by default. However, you can use Visual Basic or VBA

to switch it on again using appropriate calls to CB.Simulation or CB.RunPrefsND. Even if the Control Panel dialog is displayed, though, the Run, Analyze, and Help menus are hidden.

Your application can use any of the Crystal Ball features accessible through the Developer Kit. If users have appropriate licenses, you can also include Predictor time-series analyses in your applications and users can run simulations in Extreme speed.

Unless otherwise specified, defaults are pulled from the user's preference files. These defaults can be set through Crystal Ball's preference dialogs or preferences macro calls (such as CB.RunPrefsND) in the Crystal Ball Developer Kit.

For more information, see Chapter 5, "Crystal Ball Runtime.".

CB.RuntimeMode Example

This example hides the Crystal Ball toolbar and menus, resets the current simulation, and runs a simulation of 5000 trials.

```
CB.RuntimeMode
CB.ResetND
CB.Simulation 5000
```

CB.RuntimeMode Related Call

Name	Description
CB.Simulation	Runs a Crystal Ball simulation

CB.SaveResults

This subroutine opens the Save Results dialog so you can save results data and charts for the current simulation. To run this subroutine, you must first run a simulation, or the dialog does not open. The simulation must be completed or stopped. Once you have saved a results file with .cbr extension, you can use CB.RestoreResults or CB.RestoreResultsND to open it and restore the results.

CB.SaveResults is similar to CB.SaveRun, available in Crystal Ball 2000.5 (5.5). However, CB.SaveRun stored an entire simulation, not just the results. CB.RestoreRun and CB.RestoreRunND were used to restore the saved simulation. The existing simulation had to be reset before the saved simulation could be loaded and only one simulation could be restored at a time. This differs from the current requirements for CB.RestoreResults and CB.RestoreResultsND, discussed beginning in "CB.RestoreResults" on page 185.

See the *Oracle Crystal Ball User's Guide* for more information on saving simulation results.

Note: Before you save run results, run a simulation.

CB.SaveResults Example

This example runs a simulation of 1000 trials, and then opens the Save Results dialog.

```
CB.Simulation 1000
CB.SaveResults
```

CB.SaveResults Related Calls

Name	Description
CB.RestoreResults	Restores simulation results from a file using a dialog
CB.RestoreResultsND	Restores simulation results from a file without a dialog
CB.SaveResultsND	Saves current simulation results without a dialog
CB.Simulation	Runs a Crystal Ball simulation

CB.SaveResultsND

This subroutine automatically saves results data and charts for the current simulation into the file specified by *FileName*. Once you have saved a results file with .cbr extension, you can use CB.RestoreResults or CB.RestoreResultsND to open it and restore the results.

CB.SaveResultsND is similar to CB.SaveRunND, available in Crystal Ball 2000.5 (5.5). However, CB.SaveRunND stored an entire simulation, not just the results. CB.RestoreRun and CB.RestoreRunND were used to restore the saved simulation. The existing simulation had to be reset before the saved simulation could be loaded and only one simulation could be restored at a time. This differs from the current requirements for CB.RestoreResults and CB.RestoreResultsND, discussed beginning in "CB.RestoreResults" on page 185.

Table 155 CB.SaveResultsND Parameters

Parameter	VBA Data Type	Value	Description
FileName	String	A path surrounded by quotes (""), such as "C:\temp \sheet2.cbr"	Points to the results file where simulation results are saved

Before running this subroutine, run a simulation. The simulation must be completed or stopped but not reset.

See the *Oracle Crystal Ball User's Guide* for more information on saving simulation results.

CB.SaveResultsND Example

This example runs a simulation of 1000 trials and then saves the simulation results to file Saveme.cbr.

```
CB.Simulation 1000
CB.SaveResultsND "C:\temp\Saveme.cbr"
```

CB.SaveResultsND Related Calls

Name	Description
CB.RestoreResults	Restores simulation results from a file using a dialog
CB.RestoreResultsND	Restores simulation results from a file without a dialog
CB.SaveResults	Saves current simulation results using a dialog
CB.Simulation	Runs a Crystal Ball simulation

CB.ScatterPrefs

This call pens the Scatter Preferences dialog for the selected scatter chart, either open or closed. If more than one scatter chart is found, this call opens a chooser dialog. If no scatter chart has been defined, this call creates a new chart with no selected assumptions or forecasts using the default preferences. This chart then becomes the selected chart for subsequent scatter chart calls. If this call is used and a simulation has not been run, the macro returns error -5009.

CB.ScatterPrefs Example

The first call in this example runs a simulation. The second call either creates a scatter chart, selects a single scatter chart, or opens a chooser dialog if there are multiple scatter charts, and then opens the Scatter Preferences dialog for the selected scatter chart.

```
CB.Simulation 1000
CB.ScatterPrefs
```

CB.ScatterPrefs Related Call

Name	Description
CB.ScatterPrefsND	Sets preferences for the specified scatter chart

CB.ScatterPrefsND

This call sets preferences for the selected scatter chart. If more than one scatter chart is open and no chart is selected, this call returns a cbErrNoSelection error for CB.MacroResult.

Table 156 Parameters for CB.ScatterPrefsND

Parameter	VBA Data Type	Description
Index	Integer	Works with *Value* to set scatter chart preferences. For named constant and index values, see Table 157.
Value	Integer	Works with *Index* to set scatter chart preferences. For named constant and index values, see Table 158.

Table 157, following, lists named constant and index values for the *Index* parameter.

Table 157 CB.ScatterPrefsND Index Parameter Values — Required, Integer

Named Constant Value	Index Value	Description
cbScatterTitle	1	Defines the scatter title according to the supplied string value (title text)
cbScatterManageFore	2	Adds or removes the selected forecast to or from the chart, according to *Value*
cbScatterManageAssum	3	Adds or removes the selected assumption to or from the chart, according to *Value*
cbScatterSetTarget	5	If True, sets the target value required for Scatter View; if False, clears it
cbScatterViewType	6	Specifies the scatter view type according to *Value*
cbScatterPlotSize	7	Specifies the scatter plot size with a value from 1-10; 1 is the smallest plot and 10 is the largest
cbScatterDrawPoints	8	If True, displays scatter points; if False, does not display them
cbScatterPointSymbol	9	Specifies the scatter point symbol, according to *Value*
cbScatterPointSize	10	Specifies the size of scatter points in pixels (1-30)
cbScatterPointColor	11	Specifies the scatter point color (0-15 as defined in the Chart Preferences dialog, Chart Type tab; 0 = Auto)
cbScatterDrawLines	12	If True, displays a fit line on the scatter chart; if False, lines do not appear
cbScatterLineWidth	13	Specifies the scatter line width in pixels (1-10)
cbScatterLineType	14	Specifies the type of scatter line, according to *Value*
cbScatterLineColor	15	Specifies the scatter line color (0-15 as defined in the Chart Preferences dialog, Chart Type tab; 0 = Auto)
cbScatterCriteriaLimit	16	Specifes the scatter chart criteria to use, according to *Value*
cbScatterCriteriaLimitNumber	17	Defines the number of trials to appear, taken into account only if cbScatterCriteriaLimit = LimitTrialNumber (*Value*)
cbScatterCriteriaLimitPercentage	18	Defines the percentage of trials to show, taken into account only if cbScatterCriteriaLimit = LimitTrialPercentage (*Value*) with a possible range of 1-100

Named Constant Value	Index Value	Description
cbScatterDrawCorrelationCoefficient	19	If True, displays a correlation coefficient for each plot; if False does not display the correlation
cbScatterShowFilteredOutValues	20	If True, displays filtered out values; if False, does not display filtered out values; boolean
cbScatterShowLinearRegressionCoefficients	21	If True, displays the slope and Y-intercept for the linear regression line on each plot; if False does not display this information

Table 158, following, lists named constant and index values for the *Value* parameter.

Table 158 CB.ScatterPrefsND Value Parameter Values — Required, Integer

Used With Specified Values of Index	Named Constant Value	Index Value	Description
For *Index* 2, 3: Integer	cbScatterAddSelectedVar	1	Adds selected variable to scatter chart
For *Index* 2, 3: Integer	cbScatterRemoveSelectedVar	2	Removes selected variable from scatter chart
For *Index* 2, 3: Integer	cbScatterRemoveAllVars	3	Removes all variables from scatter chart
For *Index* 6: Integer	cbScatterView	1	Plots each selected assumption or forecast against a target (specified with *Index* =5)
For *Index* 6: Integer	cbMatrixView	2	Plots each selected assumption or forecast against every other selected variable
For *Index* 9: Integer	cbScatterPointSymbolRectangle	1	Draws rectangular scatter points
For *Index* 9: Integer	cbScatterPointSymbolCircle	2	Draws circular scatter points
For *Index* 9: Integer	cbScatterPointSymbolTriangle	3	Draws triangular scatter points
For *Index* 9: Integer	cbScatterPointSymbolDiamond	4	Draws diamond-shaped scatter points
For *Index* 9: Integer	cbScatterPointSymbolMarble	5	Draws spherical scatter points (default)
For *Index* 9: Integer	cbScatterPointSymbolHorizontalLine	6	Uses a horizontal line for scatter points
For *Index* 9: Integer	cbScatterPointSymbolVerticalLine	7	Uses a vertical line for scatter points
For *Index* 9: Integer	cbScatterPointSymbolCross	8	Uses a crossed line (+) for scatter points
For *Index* 9: Integer	cbScatterPointSymbolInvertedTriangle	9	Uses an inverted triangle for scatter points
For *Index* 14: Integer	cbScatterOrderedFit	1	Specifies an ordered fit line
For *Index* 14: Integer	cbScatterRegression	2	Specifies a linear regression line
For *Index* 16: Integer	cbScatterLimitAuto	1	Automatically limits the number of trials based on plot size
For *Index* 16: Integer	cbScatterLimitTrialNumber	2	Shows only a fixed number of trials defined through *Index* 17, cbScatterCriteriaLimitNumber

Used With Specified Values of Index	Named Constant Value	Index Value	Description
For *Index* 16: Integer	cbScatterLimitTrialPercentage	3	Shows only the specified percentage of trials defined through *Index* 18, cbScatterCriteriaLimitPercentage

CB.ScatterPrefsND Example

The following example runs a simulation with 10,000 trials and then performs the described operations to define and open a scatter chart.

```
CB.Simulation 10000

' The next command removes all variables from a chart
CB.ScatterPrefsND cbScatterManageFore, cbScatterRemoveAllVars

' The next command adds forecast from D7 to the scatter chart as a target
Range("D7").Select
CB.ScatterPrefsND cbScatterManageFore, cbScatterAddSelectedVar
CB.ScatterPrefsND cbScatterSetTarget, True

' The next commands add assumptions D4 through D6 to the scatter chart
Range("D4").Select
CB.ScatterPrefsND cbScatterManageAssum, cbScatterAddSelectedVar
Range("D5").Select
CB.ScatterPrefsND cbScatterManageAssum, cbScatterAddSelectedVar
Range("D6").Select
CB.ScatterPrefsND cbScatterManageAssum, cbScatterAddSelectedVar

' The next command sets the scatter chart title
CB.ScatterPrefsND cbScatterTitle, "Piston Displacement Scatter View"

' The next command makes sure the scatter chart is in Scatter view
CB.ScatterPrefsND cbScatterViewType, cbScatterView

' The next command sets minimal plot size (valid range is 1-10)
CB.ScatterPrefsND cbScatterPlotSize, 1

' The next command sets Criteria Limit to Auto
' (displays number of plots appropriate to plot size)
CB.ScatterPrefsND cbScatterCriteriaLimit, cbScatterLimitAuto

' The next commands specify properties of the plot points (triangle,
' size = 4, color = light green)
CB.ScatterPrefsND cbScatterDrawPoints, True
CB.ScatterPrefsND cbScatterPointSymbol, cbScatterPointSymbolTriangle
CB.ScatterPrefsND cbScatterPointSize, 4
CB.ScatterPrefsND cbScatterPointColor, 2

' The next commands draw a linear regression line on top of points
' (width = 2, color= purple)
CB.ScatterPrefsND cbScatterDrawLines, True
CB.ScatterPrefsND cbScatterLineType, cbScatterRegression
CB.ScatterPrefsND cbScatterLineWidth, 2
CB.ScatterPrefsND cbScatterLineColor, 5
```

```
' The next commands draw the correlation coefficient in each plot,
' but do not show the regression properties
CB.ScatterPrefsND cbScatterDrawCorrelationCoefficient, True
CB.ScatterPrefsND cbScatterShowLinearRegressionCoefficients, False

' The next command opens the chart
CB.OpenChart cbChtScatter
```

CB.ScatterPrefsND Related Call

Name	Description
CB.ScatterPrefs	Opens the Scatter Preferences dialog for the selected scatter chart

CB.ScenarioAnalysis

The Scenario Analysis tool runs a simulation and then sorts and matches all the resulting values of a target forecast with their corresponding assumption values. Then, you can investigate which combination of assumption values gives a particular result. You can run Scenario Analysis on any Crystal Ball model with at least one assumption and forecast that are not frozen. You select a target forecast to analyze and then the forecast's percentile or value range to examine. The resulting table shows all the values for the target forecast in the designated range, sorted, along with the corresponding assumption values.

CB.ScenarioAnalysis opens the Scenario Analysis wizard so you can run the tool.

CB.ScenarioAnalysis Example

This example resets the Crystal Ball simulation and opens the Scenario Analysis wizard.

```
CB.ResetND
CB.ScenarioAnalysis
```

CB.ScenarioAnalysis Related Calls

Name	Description
CB.GetScenarioAnalysisOption	Accesses the options set by ScenarioAnalysisND or the Scenario Analysis tool wizard
CB.ScenarioAnalysisND	Runs the Scenario Analysis tool without using dialogs

CB.ScenarioAnalysisND

The Scenario Analysis tool runs a simulation and then sorts and matches all the resulting values of a target forecast with their corresponding assumption values. Then, you can investigate which

combination of assumption values gives a particular result. You can run Scenario Analysis on any Crystal Ball model with at least one assumption and forecast that are not frozen. You select a target forecast to analyze and then the forecast's percentile or value range to examine. The resulting table shows all the values for the target forecast in the designated range, sorted, along with the corresponding assumption values.

CB.ScenarioAnalysisND runs the Scenario Analysis tool and displays results without using dialogs.

Table 159 CB.ScenarioAnalysisND Parameters

Parameter	Required?	VBA Data Type	Table
Index	Required	Integer	Table 160, "CB.ScenarioAnalysisND Index Parameter — Required, Integer," on page 199
Value1	Optional	Variant	Table 161, "CB.ScenarioAnalysisND Value1 Parameter — Optional, Variant," on page 200
Value2	Optional	Variant	Table 162, "CB.ScenarioAnalysisND Value2 Parameter — Optional, Variant," on page 200

Table 160 CB.ScenarioAnalysisND Index Parameter — Required, Integer

Named Constant Value	VBA Data Type	Index Value	Description
cbSceChooseTarget	Integer	1	Selects the target forecast to be analyzed. The range can either be specified as a separate line preceding the CB.ScenarioAnalysisND call, such as [A1:C100]. Select, or it can be the *Value1* parameter, specified as "A1:C100" or similar. If the optional *Value1* parameter is not given, the current active range is used.
cbSceRangeForecastResults	Integer	2	Works with other *Index* constants used as *Value1* to set Scenario Analysis to work with a percentile range or value range: cbSceSelectPercentiles [3] cbSceSelectForeValues [4]
cbSceSelectPercentiles	Integer	3	Works with *Value1* and *Value2* to specify a range of percentiles
cbSceSelectForeValues	Integer	4	Works with *Value1* and *Value2* to specify a range of forecast values
cbSceForeOption	Integer	5	Works with *Value1* to indicate which forecast charts to show while running Scenario Analysis
cbSceSimTrials	Integer	6	Sets the number of trials to run
cbSceIncludeNonTargetFore	Integer	7	When set to True, includes all forecasts in the output table; otherwise, includes only the target forecast
cbSceRun	Integer	8	Runs the Scenario Analysis tool when all settings are complete

Table 161 CB.ScenarioAnalysisND Value1 Parameter — Optional, Variant

Related Value	Named Constant Value	VBA Data Type	Index Value	Description
For *Index* = 1, cbSceChooseTarget	A cell reference with format:Range("A1:A1") or "A1"	Variant	n/a	Specifies the target forecast cell to be analyzed
For *Index* = 2, cbSceRangeForecastResults	cbSceSelectPercentiles	Integer	3	Indicates that a range of percentiles should be worked on
For *Index* = 2, cbSceRangeForecastResults	cbSceSelectForeValues	Integer	4	Indicates that a range of forecast values should be worked on
For *Index* = 3, cbSceSelectPercentiles	PercentileMin (a numeric value)	Double	n/a	Specifies the minimum value of the percentile range to work on; the maximum is specified as *Value2*
For *Index* = 4, cbSceSelectForeValues	ForeValueMin (a numeric value)	Double	n/a	Specifies the minimum value of the forecast value range to work on; the maximum is specified as *Value2*
For *Index* = 5, cbSceForeOption	cbSceShowDefinedFore	Integer	1	Sets the forecast option to show charts for all defined forecasts while Scenario Analysis runs
For *Index* = 5, cbSceForeOption	cbSceShowTargetFore	Integer	2	Sets the forecast option to show only the chart for the target forecast while Scenario Analysis runs
For *Index* = 5, cbSceForeOption	cbSceHideFore	Integer	3	Sets the forecast option to hide charts for all forecasts while Scenario Analysis runs (the fastest setting)
For *Index* = 6, cbSceSimTrials	TrialRuns (a numeric value)	Long integer	n/a	Specifies the number of trials to run for the scenario analysis
For *Index* = 7, cbSceIncludeNonTargetFore	True or False	Boolean	n/a	If set to True, includes all forecasts in the output table; if False, includes only the target forecast

Table 162 CB.ScenarioAnalysisND Value2 Parameter — Optional, Variant

Related Value	Named Constant Value	VBA Data Type	Index Value	Description
For *Index* = 3, cbSceSelectPercentiles	PercentileMax (a numeric value)	Double	n/a	Specifies the maximum value of the percentile range to work on; the minimum is specified as *Value1*
For *Index* = 4, cbSceSelectForeValues	ForeValueMax (a numeric value)	Double	n/a	Specifies the maximum value of the forecast value range to work on; the minimum is specified as *Value1*

CB.ScenarioAnalysisND Example

This example resets the Crystal Ball simulation, and then sets up and runs a scenario analysis as described by comments in the sample code:

```
CB.ResetND

' Select target forecast in cell B1
CB.ScenarioAnalysisND cbSceChooseTarget, "B1"


' Produce results based on a percentile range
CB.ScenarioAnalysisND cbSceRangeForecastResults, cbSceSelectPercentiles

' Use the range from the 50th percentile to the 75th
CB.ScenarioAnalysisND cbSceSelectPercentiles, 50, 75

' Show only the chart for the target forecast while the tool is running
CB.ScenarioAnalysisND cbSceForeOption, cbSceShowTargetFore

' Run 500 trials
CB.ScenarioAnalysisND cbSceSimTrials, 500

' Include only the target forecast in the results
CB.ScenarioAnalysisND cbSceIncludeNonTargetFore, False

' Run the tool
CB.ScenarioAnalysisND cbSceRun
```

CB.ScenarioAnalysisND Related Calls

Name	Description
CB.GetScenarioAnalysisOption	Accesses the options set by ScenarioAnalysisND or the Scenario Analysis tool wizard
CB.ScenarioAnalysis	Opens the Scenario Analysis tool wizard

CB.SelectAssum

This subroutine selects all assumptions in the current worksheet.

The desired worksheet must be active (selected) for Crystal Ball to run this command.

CB.SelectAssum Example

This example activates the Model worksheet of the Futura with OptQuest.xls workbook and selects all the assumptions in the worksheet.

```
Workbooks("Futura with OptQuest.xls").Worksheets("Model").Activate
CB.SelectAssum
```

CB.SelectAssum Related Calls

Name	Description
CB.SelectDecVar	Selects all decision variables in the current worksheet
CB.SelectFore	Selects all forecasts in the current worksheet

CB.SelectChart

This call selects or activates the current operating chart. If the specified name is not found, an error is thrown. If a name is not specified — for example, by passing an empty string as the name — and more than one chart of the specified type is found, a chooser dialog appears.

Currently, this call only works on current and not restored results.

Table 163 CB.SelectChart Parameters and Values

Parameter	VBA Data Type	Value	Description
SimChartType	Integer	cbChtOverlay = 1, cbChtScatter = 2, cbChtSensitiv = 3, cbChtTrend = 4	Specifies the type of chart you are selecting — overlay, scatter, sensitivity, or trend charts — according to the named constant and index values listed in the Value column
ChartID	String	The string (ChartID) returned from CB. EnumChart or CB.CreateChart.	Indicates the chart to select

CB.SelectChart Example

This example runs a simulation and selects the next overlay chart found in the workbook. Because CB.EnumChart is used here, call CB.CheckData.

```
CB.Simulation 1000
CB.CheckData
CB.SelectChart cbChtOverlay, CB.EnumChart(cbChtOverlay)
```

CB.SelectChart Related Calls

Name	Description
CB.CheckData	Checks decision variables for validity, initializes variables
CB.CreateChart	Creates a new chart of the specified type with the optional name, if specified
CB.EnumChart	Returns the unique name of the next chart of the specified type

CB.SelectDecVar

This subroutine selects all decision variables in the current worksheet. The desired worksheet must be active for Crystal Ball to run this command.

Note: Before calling this subroutine, activate the worksheet where you want to select Crystal Ball data.

CB.SelectDecVar Example

This example activates the Model worksheet of the Futura with OptQuest.xls workbook and selects all decision variables in the worksheet.

```
Workbooks("Futura with OptQuest.xls").Worksheets("Model").Activate
CB.SelectDecVar
```

CB.SelectDecVar Related Calls

Name	Description
CB.SelectAssum	Selects all assumptions in the current worksheet
CB.SelectFore	Selects all forecasts in the current worksheet

CB.SelectFore

This subroutine selects all forecasts in the current worksheet.

Before calling this subroutine, activate the worksheet where you want to selectCrystal Ball data.

CB.SelectFore Example

This example activates the Model worksheet of the Futura with OptQuest.xls workbook and selects all the forecasts on the worksheet.

```
Workbooks("Futura with OptQuest.xls").Worksheets("Model").Activate
CB.SelectFore
```

CB.SelectFore Related Calls

Name	Description
CB.SelectAssum	Selects all assumptions in the current worksheet
CB.SelectDecVar	Selects all decision variables in the current worksheet

CB.SensPrefs

Opens the Sensitivity Preferences dialog for the selected sensitivity chart, either open or closed. If more than one sensitivity chart is found, opens a chooser dialog. If no sensitivity chart has been defined, this call creates a new chart with no associated forecast using the default preferences. This chart then becomes the selected chart for subsequent sensitivity chart calls. If this call is used and a simulation has not been run, the macro returns error -5009.

CB.SensPrefs Example

This example runs a simulation with 1000 trials, either creates a sensitivity chart, selects a single sensitivity chart, or opens a chooser dialog if there are multiple sensitivity charts, and then opens the Sensitivity Preferences dialog for the selected sensitivity chart.

```
CB.Simulation 1000
CB.SensPrefs
```

CB.SensPrefs Related Call

Name	Description
CB.SensPrefsND	Sets preferences for the specified sensitivity chart without using a dialog

CB.SensPrefsND

The function sets preferences for the selected sensitivity chart without using a dialog. If more than one sensitivity chart is open and no chart is selected, this call returns a cbErrNoSelection error for CB.MacroResult.

Table 164 Parameters for CB.SensPrefsND

Parameter	VBA Data Type	Value	Description
Index	Integer	See Table 165	Works with optional parameters *Value1* and *Value2* to set sensitivity chart preferences
Value1 (Optional)	Variant	See Table 166	Works with the *Index* parameter to set sensitivity chart preferences
Value2 (Optional)	Variant	See Table 167	Works with *Index* = 5 or 6 to set the maximum or minimum number of sensitivity values

Table 165, following, lists named constant and index values for the *Index* parameter.

Table 165 CB.SensPrefsND Index Parameter Values — Required, Integer

Named Constant Value	Index Value	Description
cbSenTargetFore	1	Defines the selected forecast cell as the target forecast

Named Constant Value	Index Value	Description
cbSenMeasure	2	Sets whether to measure sensitivity by rank correlation or by contribution to variance, according to *Value1*
cbSenDisplayOnlyHighest	5	Sets to display only the highest sensitivities, according to *Value1* and *Value2*
cbSenDisplayGreaterThan	6	Sets to display only sensitivities greater than *Value2*, according to *Value1*
cbSenTitle	7	Gives the sensitivity chart a title according to *Value1*
cbSenChooseAssum	8	Clears and selects the assumptions for the sensitivity chart, according to *Value1*

Note: *Index* values cbSenIncludeAssum [3] and cbSenIncludeOtherFore [4] are obsolete.

Table 166, following, lists values for the optional *Value1* parameter.

Table 166 CB.SensPrefsND Value1 Parameter Values — Optional, Variant

Used With Specified Values of Index	Named Constant or Boolean Value	Index Value	Description
For *Index* = 2: Boolean	True or False OR cbMesRankCorrelation or cbMesContributionToVar	n/a	True measures sensitivity by rank correlation. False uses contribution to variance.
For *Index* = 5: Boolean	True or False	n/a	True displays only a certain number (*Value2*) of sensitivities. False does not.
For *Index* = 6: Boolean	True or False	n/a	True only displays sensitivities greater than *Value2*
For *Index* = 7: String	Title in quotes ("")	n/a	Defines the sensitivity chart title
For *Index* = 8: Integer	cbChaAll	1	Includes all assumptions
For *Index* = 8: Integer	cbChaAdd	3	Adds the assumption in the selected cell to the list of chosen assumption
For *Index* = 8: Integer	cbChaClearList	4	Clears the list of chosen assumptions
For *Index* = 8: Integer	cbChaRemove	6	Removes the assumption in the selected cell from the list of chosen assumptions

Table 167, following, lists values for the optional *Value2* parameter.

Table 167 CB.SensPrefsND Value2 Parameter Values — Optional, Variant

Used With Specified Values of Index	Value	Description
For *Index* = 5: Integer	A positive, whole number	Sets the maximum number of sensitivities to display
For *Index* = 6: Double	A positive number between 0.00 and 1.00	Sets the minimum sensitivity values to display

CB.SensPrefsND Example

The following example runs a simulation, selects cell B2, defines that cell as the target forecast for a sensitivity chart, creates and names the sensitivity chart, indicates the sensitivity should be shown as Contribution to Variance, limits display to the four most sensitive assumptions with sensitivities greater than 0.1, and finally opens the chart.

```
CB.Simulation 1000
Range("B2").Select
CB.SensPrefsND cbSenTargetFore
CB.SensPrefsND cbSenTitle, "My Sensitivity Chart"
CB.SensPrefsND cbSenMeasure, cbMesContributionToVar
CB.SensPrefsND cbSenDisplayOnlyHighest, True, 4
CB.SensPrefsND cbSenDisplayGreaterThan, True, 0.1
CB.OpenSensitiv
```

CB.SensPrefsND Related Call

Name	Description
CB.SensPrefs	Opens the Sensitivity Preferences dialog for the selected sensitivity chart

CB.SetAssum

This function changes attributes for an assumption without redefining the assumption using CB.DefineAssumND.

Before you call this function, the assumption must exist.

Table 168 CB.SetAssum Returned Data Type

Returned Value	Returned Data Type
Whether the function was successful	Variant

Table 169 CB.SetAssum Parameters

Parameter	VBA Data Type	Value	Description
AssumReference	Variant	In a macro: Range("B3")	Points to the assumption cell with the data to change
Index	Integer	See Table 170	Changes the assumption name and truncation points according to *Value*.
Value	Variant	See Table 171	Works with *Index* to specify an assumption name and truncation values

Table 170, following, describes the *Index* parameter.

Table 170 CB.SetAssum Index Parameter Values — Required, Integer

Named Constant Value	Index Value	Description
cbAsmName	1	Changes the assumption name according to *Value*
cbAsmLowCutOff	6	Changes the lower truncation point according to *Value*
cbAsmHighCutOff	7	Changes the higher truncation point according to *Value*

Table 171, following, describes the *Value* parameter.

Table 171 CB.SetAssum Value Parameter Values — Required, Variant

Used With Specified Values of Index	Value	Description
For *Index* = cbAsmName [1]: String	Assumption name, in quotes ("")	Specifies the new assumption name
For *Index* = cbAsmLowCutOff [6]: Double	Any number	Sets the new lower truncation point
For *Index* = cbAsmHighCutOff [7]: Double	Any number	Sets the new upper truncation point

CB.SetAssum Example

This example changes the name of the assumption in cell B3 to "Revised Assumption."

```
CB.SetAssum Range("B3"), 1, "Revised Assumption"
```

CB.SetAssum Related Calls

Name	Description
CB.DefineAssumND	Defines or changes assumptions in selected cells without using a dialog
CB.GetAssum	Retrieves information for a specific assumption cell

CB.SetCBAutoLoad

This call specifies whether to automatically start Crystal Ball whenever Microsoft Excel starts, called "autoloading."

The original default setting when Crystal Ball is first installed is False. However, the following alternatives can be used to activate autoloading using dialogs instead of the CB.SetCBAutoLoad call:

- Start the Crystal Ball Application Manager and check or uncheck "Automatically launch Crystal Ball when Microsoft Excel starts."
- Open the Microsoft Excel COM Add-ins dialog and check or uncheck Crystal Ball.

For more information on these alternatives, see "Starting Crystal Ball" in the current *Crystal Ball Installation and Licensing Guide*.

No matter which alternative you use, the same setting appears in either of the other two alternatives (including the CB.SetCBAutoLoad call). The autoload setting is persistent; once set, it remains set until one of the alternatives is used to change it.

Table 172 CB.SetCBAutoLoad Parameter

Parameter	VBA Data Type	Named Constant Value	Description
autoLoad	Boolean	True or False	True sets Crystal Ball to autoload each time Microsoft Excel starts. False deactivates autoloading.

CB.SetCBAutoLoad Example

This code sets Crystal Ball to automatically start when Microsoft Excel starts.

```
CB.SetCBAutoLoad(True)
```

CB.SetCBAutoLoad Related Calls

Name	Description
CB.CBLoaded	Indicates whether Crystal Ball is currently loaded within Microsoft Excel
CB.GetCBAutoLoad	Returns the current Crystal Ball autoload setting that indicates whether Crystal Ball is set to open whenever Microsoft Excel opens
CB.Shutdown	Closes Crystal Ball while leaving Microsoft Excel open
CB.Startup	Loads Crystal Ball into Microsoft Excel if possible and indicates when Crystal Ball has been successfully opened

CB.SetCBWorkbookPriority

This subroutine sets the running order for Crystal Ball macros when multiple workbooks are open. For example, a workbook with *aPriority* = 10 will run before a workbook with *aPriority* = 20. CB.SetCBWorkbookPriority runs only at the workbook level and not at the worksheet level. For details, see “Running user-defined macros” at the end of Chapter 5 in the *Oracle Crystal Ball User's Guide*.

Table 173 CB.SetCBWorkbookPriority Parameter

Parameter	VBA Data Type	Value	Description
aPriority	Long	An integer of any size, usually positive	Specifies the running priority of user-defined macros in a workbook. The smaller the positive value, the higher the priority.

CB.SetCBWorkbookPriority Example

This example assigns a priority of 10 to the workbook Book1.xls.

```
Application.Workbooks("Book1.xls").Activate
CB.SetCBWorkbookPriority(10)
```

CB.SetDecVar

This function changes certain attributes for a decision variable without redefining the decision variable using CB.DefineDecVarND.

Table 174 CB.SetDecVar Returned Data Type

Returned Value	Returned Data Type
Whether the function was successful	Variant

Before you call this function, the decision variable must exist and the simulation must be reset.

Table 175 CB.SetDecVar Parameters

Parameter	VBA Data Type	Value	Description
DecVarReference	Variant	In a macro: Range("B3")	Points to the decision variable cell with the data to change
Index	Integer	See Table 176	Works with *Value* to set decision variable attributes
Value	Variant	See Table 177	Changes decision variable attributes specified with *Index*

Table 176, following, describes the *Index* parameter.

Table 176 CB.SetDecVar Index Parameter Values — Required, Integer

Named Constant Value	Index Value	Description
cbDecName	1	Changes the decision variable name according to *Value*
cbDecMin	2	Changes the minimum (lower) bound, according to *Value*
cbDecMax	3	Changes the maximum (upper) bound, according to *Value*
cbDecType	4	Changes the decision variable type, according to *Value*
cbDecStep	5	Changes the step size for discrete decision variables, depending on *Value*.
cbDecFrozen	6	Freezes the selected decision variable, depending on *Value*
cbDecCustomData	7	Supplies a list of possible values for cbDecType = Custom [4]. Decimals can be used

Table 177, following, describes the *Value* parameter.

Table 177 CB.SetDecVar Value Parameter Values — Required, Variant

Used With Specified Values of Index	Value	Description
For *Index* = 1, cbDecName: String	Decision variable name, in quotes ("")	Specifies the new decision variable name
For *Index* = 2, cbDecMin: Double	Any number	Sets the new minimum (lower) bound
For *Index* = 3, cbDecMax: Double	Any number	Sets the new maximum (upper) bound
For *Index* = 4, cbDecType: Integer	cbDecVarTypeContinuous = 0, cbDecVarTypeDiscrete = 1, cbDecVarTypeBinary = 2, cbDecVarTypeCategory = 3, cbDecVarTypeCustom = 4	For a detailed description of each of these cbDecType settings, see "CB.DefineDecVarND" on page 108. For cbDecVarTypeCustom, a string of data must be supplied with Index = cbDecCustomData [7].
For *Index* = 5, cbDecStep: Double	Any whole number smaller than the range and greater than 0	Sets the new step size for discrete variables
For *Index* = 6, cbDecFrozen: Boolean	True or False	True freezes the current decision variable. False unfreezes it, if frozen
For *Index* = 7, cbDecCustomData: String	A list of at least two values separated by commas (,) or a reference to a range of cells containing these values; decimal values can be used. The range must be one-dimensional (adjoining cells in one row *or* one column).	Supplies a list of data for cbDecType = cbDecVarTypeCustom [4]

CB.SetDecVar Example 1

This example changes the lower bound of the decision variable in cell B4 to 0.

```
CB.SetDecVar Range("B4"), 2, 0
```

CB.SetDecVar Example 2

This example changes the type of the decision variable in cell B4 to Custom and passes the following values to it: 10, 20, 30, 40.

```
CB.SetDecVar Range("B4"), cbDecType, cbDecVarTypeCustom
CB.SetDecVar Range("B4"), cbDecCustomData, "10, 20, 30, 40"
```

CB.SetDecVar Related Calls

Name	Description
CB.DefineDecVar	Defines or changes a decision variable in a selected cell using a dialog
CB.DefineDecVarND	Defines or changes a decision variable in a selected cell without a dialog

Name	Description
CB.GetDecVar	Retrieves information for a specific decision variable cell

CB.SetExcel2007ForegroundMode

This function is used in Microsoft Excel 2007 to set the Normal Speed preference "Bring Microsoft Excel 2007 to the foreground while running Normal speed simulations (faster)." When set to True, this call runs simulations at optimal speed when using Normal speed in Microsoft Excel 2007. This call is equivalent to setting the referenced preference manually in the Normal Speed options dialog, accessed by selecting Run Preferences, then Speed, then Normal Speed, and then Options.

Table 178 CB.SetExcel2007ForegroundMode Parameter

Parameter	VBA Data Type	Value	Description
Enable	Boolean	True or False	Indicates whether to run Microsoft Excel 2007 in the foreground when using Normal speed. True is equivalent to manually checking the preference in the Normal Speed options dialog; False is equivalent to manually unchecking the preference.

CB.SetExcel2007ForegroundMode Example

This example unchecks, or clears, the preference "Bring Microsoft Excel 2007 to the foreground while running Normal speed simulations (faster)" in the Normal Speed options dialog, accessed manually by choosing Normal Speed on the Speed tab of the Run Preferences dialog and then clicking the Options button. When set to False, simulations in Microsoft Excel 2007 might run more slowly than if the preference were set to True (the default).

```
CB.SetExcel2007ForegroundMode False
```

CB.SetExcel2007ForegroundMode Related Call

Name	Description
CB.GetExcel2007ForegroundMode	Returns the current Run Preferences foreground mode setting for Microsoft Excel 2007 in Normal speed

CB.SetFitRange

This function is used with CB.Fit ("CB.Fit" on page 127) to specify a worksheet range that contains the set of data to fit to a distribution.

In a subroutine, this call takes the form:

```
CB.SetFitRange Range("A1:A15")
```

Table 179 CB.SetFitRange Parameter

Parameter	VBA Data Type	Value	Description
RangeRef	Variant	In a macro: Range("B3")	Points to the cell range with the data to fit to a distribution

CB.SetFitRange Examples

See the examples for "CB.Fit" on page 127.

CB.SetFitRange Related Call

Name	Description
CB.Fit	Fits a specific probability distribution to a range of data and returns information about the fit

CB.SetFore

This function changes forecast attributes without redefining the forecast using CB.DefineForeND.

Before you call this function, the forecast must exist.

Table 180 CB.SetFore Returned Data Type

Returned Value	Returned Data Type
Whether the function was successful	Variant

Table 181 CB.SetFore Parameters

Parameter	VBA Data Type	Value	Description
ForeReference	Variant	In a macro: Range("B3")	Points to the forecast cell with the desired data
Index	Integer	See Table 182	Sets certain forecast attributes, according to *Value*
Value	Variant	See Table 183	Works with *Index* to set certain forecast attributes

Table 182, following, describes the *Index* parameter.

Table 182 CB.SetFore Index Parameter Values — Required, Integer

Named Constant Value	Index Value	Description
cbForName	1	Sets the forecast name, according to *Value*
cbForUnits	2	Sets the forecast units, according to *Value*

Named Constant Value	Index Value	Description
cbForAutoOpen	4	Changes whether Crystal Ball displays the forecast window automatically, according to *Value*
cbForOpenWhen	5	Changes when Crystal Ball displays the forecast window, according to *Value*
cbForPrecision	6	Changes whether Crystal Ball uses precision of the forecast value to stop a simulation, according to *Value*
cbForPrecisionAbsolute	7	Changes whether Crystal Ball uses absolute or relative terms for the size of the confidence interval, according to *Value*
cbForPrecisionAbsoluteValue	8	Sets the size of the confidence interval in forecast units, according to *Value*
cbForPrecisionRelativeValue	9	Sets the size of the confidence interval in percentage terms, according to *Value*
cbForPrecisionMean	10	Sets whether Crystal Ball uses the precision of the mean as a stopping criterion, according to *Value*
cbForPrecisionStdDev	11	Sets whether Crystal Ball uses the precision of the standard deviation as a stopping criterion, according to *Value*
cbForPrecisionPerc	12	Sets whether Crystal Ball uses the precision of a percentile value as a stopping criterion, according to *Value*
cbForPrecisionPercValue	13	Sets the percentile value that Crystal Ball checks for the indicated precision, according to *Value*
cbForFilter	14	Sets whether Crystal Ball filters forecast values, according to *Value*
cbForFilterInside	15	Sets whether discards forecast values inside or outside the indicated range, according to *Value*
cbForFilterFrom	16	Sets the lower bound of the filtering range, according to *Value*
cbForFilterTo	17	Sets the upper bound of the filtering range, according to *Value*
cbForFilterGlobal	18	Specifies whether Crystal Ball applies the filtering options to equivalent trials of all forecasts, according to *Value*
cbForFrozen	19	Freezes the forecast, according to *Value*
cbForLSL	20	Enters a lower specification limit for the forecast, according to *Value*
cbForUSL	21	Enters an upper specification limit for the forecast, according to *Value*
cbForTarget	22	Enters a target value for the forecast, according to *Value*

Note: *Index* = cbForWindowSize = 3 is obsolete and is not included here.

Table 183, following, describes the *Value* parameter.

Table 183 CB.SetFore Value Parameter Values — Required, Variant

Used With Specified Values of Index	Value	Description
For *Index* = 1, cbForName: String	Forecast name, in quotes ("")	Specifies the new forecast name
For *Index* = 2, cbForUnits: String	Units name, in quotes ("")	Specifies the new units
For *Index* = 4, cbForAutoOpen: Boolean	True or False	True displays the window automatically. False does not.
For *Index* = 5, cbForOpenWhen: Integer	0 or 1: cbOpnWhenRunning = 0, cbOpnWhenStopped = 1	Sets when to display the forecast chart: 0 = during the simulation; 1 = after the simulation stops
For *Index* = 6, cbForPrecision: Boolean	True or False	True turns on precision control. False turns it off.
For *Index* = 7, cbForPrecisionAbsolute: Boolean	True or False	True sets the confidence interval to be in absolute units. False sets it in relative terms.
For *Index* = 8, cbForPrecisionAbsoluteValue: Double	Units	Sets the size of the confidence interval in terms of forecast units
For *Index* = 9, cbForPrecisionRelativeValue: Double	0.01 to 99.99	Sets the size of the confidence interval in (relative) percentage terms
For *Index* = 10, cbForPrecisionMean: Boolean	True or False	True checks the precision of the mean. False does not.
For *Index* = 11, cbForPrecisionStdDev: Boolean	True or False	True checks the precision of the standard deviation. False does not.
For *Index* = 12, cbForPrecisionPerc: Boolean	True or False	True checks the precision of a percentile. False does not.
For *Index* = 13, cbForPrecisionPercValue: Double	0.01 to 99.99	Sets the percentile value that Crystal Ball checks for precision
For *Index* = 14, cbForFilter: Boolean	True or False	True turns on forecast filtering. False turns it off.
For *Index* = 15, cbForFilterInside: Boolean	True or False	True discards values inside the filtering range. False discards values outside the range.
For *Index* = 16, cbForFilterFrom: Double	Units	Sets the lower bound of the filtering range
For *Index* = 17, cbForFilterTo: Double	Units	Sets the upper bound of the filtering range
For *Index* = 18, cbForFilterGlobal: Boolean	True or False	True applies the filtering settings to equivalent trials for all forecasts in a simulation. False applies them only to the current forecast.
For *Index* = 19, cbForFrozen: Boolean	True or False	True freezes the forecast so it will not recalculate when a simulation runs. False unfreezes the forecast.
For *Index* = 20, cbForLSL: Variant	A numeric value (double), a string representing a cell reference (such as "=B1"), or cbParmUndefined	Sets the lower specification limit (LSL) for the forecast

Used With Specified Values of Index	Value	Description
For *Index* = 21, cbForUSL: Variant	A numeric value (double), a string representing a cell reference (such as "=B1"), or cbParmUndefined	Sets the upper specification limit (USL) for the forecast
For *Index* = 22, cbForTarget: Variant	A numeric value (double), a string representing a cell reference (such as "=B1"), or cbParmUndefined	Sets the target value for the forecast

CB.SetFore Example 1

This example changes the forecast in cell B7 to open the forecast window automatically during the simulation.

```
CB.SetFore Range("B7"), 4, True
CB.SetFore Range("B7"), 5, 0
```

CB.SetFore Example 2

This example turns on precision control for the forecast in cell B7, selects the absolute size of the confidence interval, sets the size at 10 forecast units, and selects the use of the mean in checking precision.

```
CB.SetFore Range("B7"), 6, True
CB.SetFore Range("B7"), 7, True
CB.SetFore Range("B7"), 8, 10
CB.SetFore Range("B7"), 10, True
```

CB.SetFore Related Calls

Name	Description
CB.DefineForeND	Defines forecasts in selected cells without a dialog
CB.GetFore	Retrieves information for a specific forecast cell
CB.SetFore	Changes the settings for a forecast

CB.SetRange

This subroutine sets the certainty range for one or all forecasts.

Note: Run a simulation before calling this subroutine; otherwise CB.MacroResult returns cbErrNotReady. To apply the subroutine to only one cell, select that cell.

Table 184 CB.SetRange Parameters

Parameter	VBA Data Type	Value	Description
Min (Optional)	Variant	Any number or cbMinusInfinity	Specifies the value of the low end of the certainty range and locks the certainty grabber there (if greater than –Infinity). If only *Min* is set, *Max* is set to +Infinity and the grabber is unlocked. If both *Min* and *Max* are set, both grabbers are locked.
Max (Optional)	Variant	Any number or cbPlusInfinity	Specifies the value of the high end of the certainty range and locks the certainty grabber there (if less than +Infinity). If only *Max* is set, *Min* is set to –Infinity and unlocked. If both *Min* and *Max* are set, both grabbers are locked.
Certainty (Optional)	Variant	Any number between 0 and 100	Specifies the certainty level (as a percentage) to display. If *Certainty* and *Min* are both set, *Max* is set to the appropriate value but is not locked. If *Certainty* and *Max* are both set, *Min* is set to the appropriate value but is not locked.
All (Optional)	Variant	True or False	Specifies whether to apply this subroutine to all forecasts (True) or just the selected forecast cell (False). The default is False. If this parameter is set to False and the selected cell contains no forecast, CB.MacroResult returns cbErrNoForecastInCell.

You can specify one or two of *Min*, *Max*, and *Certainty*. If you specify all three, Crystal Ball uses *Min* and *Max* to determine the certainty and ignores the entered *Certainty* value. If *Min* or *Max* is set to a value other than –Infinity or +Infinity, the certainty grabber is locked at the indicated value. If either *Min* or *Max* is set to –Infinity or +Infinity, the certainty grabber is not locked there.

CB.SetRange Example 1

This example runs a simulation of 1000 trials, selects cell B7, sets the certainty range to 51%, and applies it only to the forecast in cell B7.

```
CB.Simulation 1000
Range("B7").Select
CB.SetRange , , 51
```

CB.SetRange Example 2

This example runs a simulation of 1500 trials, sets the certainty range to between negative infinity and 1000, and applies this certainty range to all forecasts.

```
CB.Simulation 1500
CB.SetRange cbMinusInfinity, 1000, , True
```

CB.SetRange Related Call

Name	Description
CB.Simulation	Runs a Crystal Ball simulation

CB.Shutdown

If Crystal Ball is loaded into Microsoft Excel, this function closes Crystal Ball without closing Microsoft Excel. It returns True when Crystal Ball is successfully unloaded.

Table 185 CB.Shutdown Returned Data Type

Returned Value	Returned Data Type
The current Crystal Ball load status	Boolean

CB.Shutdown Example

This code unloads Crystal Ball and returns TRUE in cell B2 when unloading is complete.

```
Range("B2").Value = CB.Shutdown()
```

CB.Shutdown Related Calls

Name	Description
CB.CBLoaded	Indicates whether Crystal Ball is currently loaded within Microsoft Excel
CB.GetCBAutoLoad	Returns the current Crystal Ball autoload setting that indicates whether Crystal Ball is set to open whenever Microsoft Excel opens
CB.SetCBAutoLoad	Sets or cancels autoloading with Microsoft Excel; determines whether Crystal Ball is set to open whenever Microsoft Excel opens
CB.Startup	Loads Crystal Ball into Microsoft Excel if possible and indicates when Crystal Ball has been successfully opened

CB.SimResult

This function returns the criterion that finally stopped the most recent simulation.

Note: Before calling this function, run a simulation or the function returns cbErrNotReady.

Table 186 CB.SimResult Returned Data Type

Returned Value	Returned Data Type
The stopping criterion used: 0 for a calculation error, 1 for maximum number of trials, 2 for Crystal Ball precision control, or 3 for OptQuest confidence testing	Variant

CB.SimResult Example

This example runs a simulation. If Crystal Ball stops the simulation after the maximum number of trials (1000 in this example), CB.SimResult puts a 1 in cell B3.

```
CB.Simulation 1000
Range("B3") = CB.SimResult
```

CB.SimResult Related Calls

Name	Description
CB.RunPrefsND	Sets what stopping criterion to use without a dialog
CB.Simulation	Runs a Crystal Ball simulation

CB.Simulation

This subroutine runs a Crystal Ball simulation on any open workbooks with Crystal Ball data for a specified number of trials.

Note: The hide and minimize options are useful for increasing the speed of simulations.

Caution! Note that Crystal Ball cannot run a simulation if a cell with Crystal Ball data is located in a write-protected worksheet. If you run a simulation using CB.Simulation without resetting, Crystal Ball runs additional trials and adds the results to the existing results. This means you can run more trials than the maximum set with the Run Preferences dialog or a Developer Kit macro call.

Table 187 CB.Simulation Parameters

Parameter	VBA Data Type	Value	Description
Iterations	Long	Any positive integer	Specifies the number of trials to run. If you already ran a simulation and did not reset it, Crystal Ball runs this many more trials.
Obsolete_ Asynchronous (Optional)	Variant	null	Used as a placeholder for the Asynchronous parameter, removed starting with Crystal Ball 11.1.1.0.00.

Parameter	VBA Data Type	Value	Description
HideForecasts (Optional)	Variant	True or False	Sets whether to suppress forecast windows from opening normally during simulations (True) or to let them open as set in each forecast (False). Suppressing forecast windows can help speed up long simulations.
MinimizeHost (Optional)	Variant	True or False	Sets whether to minimize Microsoft Excel during simulations (True) or not (False). Minimizing Microsoft Excel can help speed up long simulations. This parameter only works in Normal or Demo speed, not Extreme speed.
MinimizeAllSheets (Optional)	Variant	True or False	Sets whether to minimize worksheets during simulations (True) or not (False). Minimizing worksheets can help speed up long simulations. This parameter only works in Normal or Demo speed, not Extreme speed.
StatusString (Optional)	Variant	Words surrounded by quotes ("")	Inserts a string into the status bar during the simulation. If this is omitted, no special string appears.
ShowControlPanel (Optional)	Boolean	True or False	Sets whether to show the Crystal Ball Control Panel during simulations (True) or not (False). Hiding it can help speed up long simulations. The default is False for simulations run through the Crystal Ball Developer Kit. This parameter is new for Crystal Ball versions later than 5.5.

CB.Simulation Example 1

This example resets any previously run simulation and runs a simulation of 1000 trials.

```
'Following call makes sure the trials begin at 1
CB.ResetND

CB.Simulation 1000
```

CB.Simulation Example 2

This example switches to Normal speed and runs a simulation of 100,000 trials. Because of the lengthy number of trials, all the time-saving options of suppressing forecast windows and minimizing worksheets and Microsoft Excel are set to True. This call does not set a text string to use in the Microsoft Excel status bar, since Microsoft Excel is minimized during the simulation and any string would never be displayed.

```
CB.ResetND
CB.RunPrefsND cbRunMode, cbRunNormalSpeed
CB.Simulation 100000,, True, True, True
```

CB.Simulation Related Calls

Name	Description
CB.Reset	Resets the current simulation after prompting
CB.ResetND	Resets the current simulation without confirmation
CB.StartMultiSimul	Marks the start of multiple simulations
CB.StopMultiSimul	Marks the end of multiple simulations

CB.SingleStep

This subroutine runs one trial of a simulation, changing the assumptions once.

The simulation must not be running when you use CB.SingleStep.

CB.SingleStep Example

This example resets any previously run simulation and single-steps once (runs one trial).

```
' The following call makes sure the trials begin at 1
CB.ResetND

CB.SingleStep
```

CB.SingleStep Related Call

Name	Description
CB.Simulation	Runs simulation for specified number of iterations

CB.StartMultiSimul

This subroutine enables you to use resources more effectively when running multiple simulations using CB.Simulation. It calls CheckData on the first simulation and then increments the iterations counter at the start of each simulation.

If you call CB.StartMultiSimul before multiple simulations, then call CB.StopMultiSimul after the simulations are run.

Table 188 CB.StartMultiSimul Parameters

Parameter	VBA Data Type	Value	Description
Rescan (Optional)	Boolean	True or False	Indicates whether to rescan the workbook for each simulation call within a multiple simulation. The default is False, which eliminates rescans for faster processing. This optional parameter only works in Extreme Speed.
SaveAssumValues (Optional)	Boolean	True or False	If set to True, saves assumption values from multiple simulations for use in sensitivity analysis and other analyses. The default is False.
MultiAutoDownshift (Optional)	Boolean	True or False	With CB.AutoDownshift set to True, if *MultiAutoDownshift* is set to True, Crystal Ball automatically downshifts to Normal speed when Extreme speed incompatibilities occur. No dialog appears. The default is False. This optional parameter only works in Extreme Speed.

Note: If a non-decision variable cell referenced in a forecast formula (a forecast precedent) is changed between simulations, there might be slight differences in results depending on whether *Rescan* is set to True or False. If custom code is created where precedent values are changed manually between simulations, these models need to call StartMultiSimul with *Rescan* set to True. This change might also be true for some models running in Extreme Speed.

CB.StartMultiSimul Example

This example runs 100 simulations with rescanning. The array Median is filled with the values of the medians for each simulation in the multiple simulation.

```
Dim Median(50) As Double
CB.StartMultiSimul(True)
For i = 1 To 100
   CB.ResetND
   CB.Simulation 1000
   Median(i) = CB.GetForePercent(Workbooks("Futura with OptQuest.xls"). _
      Worksheets("Model").Range("C7"), 50)
Next i
CB.StopMultiSimul
```

CB.StartMultiSimul Related Calls

Name	Description
CB.Simulation	Runs a simulation
CB.StopMultiSimul	Marks the end of multiple simulations

CB.Startup

If Crystal Ball is not loaded into Microsoft Excel, this function loads Crystal Ball. It returns True when Crystal Ball is loaded.

Table 189 CB.Startup Returned Data Type

Returned Value	Returned Data Type
The current Crystal Ball load status	Boolean

CB.Startup Example

This example loads Crystal Ball and returns TRUE in cell B2 when loading is complete.

```
Range("B2").Value = CB.Startup()
```

CB.Startup Related Calls

Name	Description
CB.CBLoaded	Indicates whether Crystal Ball is loaded in Microsoft Excel
CB.GetCBAutoLoad	Returns the Crystal Ball autoload setting that indicates whether Crystal Ball is set to open whenever Microsoft Excel opens
CB.SetCBAutoLoad	Sets or cancels autoloading with Microsoft Excel; determines whether Crystal Ball is set to open whenever Microsoft Excel opens
CB.Shutdown	Closes Crystal Ball while leaving Microsoft Excel open

CB.StopMultiSimul

This subroutine more effectively uses resources when running multiple simulations with CB.Simulation.

Call CB.StartMultiSimul before multiple simulations, and then call CB.StopMultiSimul after the simulations are run. See the example for CB.StartMultiSimul.

CB.StopMultiSimul Related Calls

Name	Description
CB.Simulation	Runs a simulation
CB.StartMultiSimul	Marks the start of multiple simulations

CB.TrendPrefs

Opens the Trend Preferences dialog for the selected trend chart, either open or closed. If more than one trend chart is found, opens a chooser dialog. If no trend chart has been defined, this call creates a new chart with no associated forecast using the default preferences. This chart then becomes the selected chart for subsequent trend chart calls. A simulation must be run before this call is used.

CB.TrendPrefs Example

This example runs a simulation, either creates a trend chart, selects a single trend chart, or opens a chooser dialog if there are multiple trend charts, and then opens the Trend Preferences dialog for the selected trend chart.

```
CB.Simulation 1000
CB.TrendPrefs
```

CB.TrendPrefs Related Calls

Name	Description
CB.TrendPrefsND	Sets preferences for the specified trend chart without using a dialog

CB.TrendPrefsND

The function sets preferences for the selected trend chart. If more than one trend chart is defined and no chart is selected, this call returns an error.

Note: Before calling CB.TrendPrefsND, run a simulation.

Table 190 Parameters for CB.TrendPrefsND

Parameter	VBA Data Type	Value	Description
Index	Integer	See Table 191	Works with the other parameters to set trend chart preferences
Value1	Variant	See Table 192	Works with the other parameters to set trend chart preferences
Value2 (Optional)	Variant	See Table 193	Works with the other parameters to set trend chart preferences
Value3 (Optional)	Variant	See Table 194	Works with the other parameters to set trend chart preferences
Value4 (Optional)	For *Index* = 8: Integer	Any integer between 0 and 100 (inclusive)	Sets a fourth certainty band to display. Use this with *Value1* through *Value3*

Parameter	VBA Data Type	Value	Description
Value5, *Value6*, *Value7* (All optional)	Integer	Any integer between 0 and 100 (inclusive)	Sets up to three more certainty bands to display. Use this with *Value1* through *Value4*

Table 191, following, describes the *Index* parameter.

Table 191 CB.TrendPrefsND Index Parameter Values — Required, Integer

Named Constant Value	Index Value	Description
cbTrdTitle	1	Gives the trend chart a title, as defined by *Value1*
cbTrdChooseFore	2	Clears and selects the forecasts for the trend chart, according to *Value1*
cbTrdVertGridLines	4	Turns on and off the vertical grid lines, according to *Value1*
cbTrdHorzGridLines	5	Turns on and off the horizontal grid lines, according to *Value1*
cbTrdChartType	6	Sets the trend chart type, according to *Value1*
cbTrdValueAxis	7	Sets options for the value axis, according to *Value1*
cbTrdCertaintyBands	8	Sets up to 7 certainty bands to display, according to *Value1* through *Value7*

Note: *Index* = cbTrdForeTitleAxis [3] is now obsolete.

Table 192, following, describes the *Value1* parameter.

Table 192 CB.TrendPrefsND Value1 Parameter Values — Required, Variant

Used With Specified Values of Index	Named Constant or Other Value	Index Value	Description
For *Index* = 1: String	Title in quotes ("")	n/a	Defines the trend chart title
For *Index* = 2: Integer	cbChfAdd	4	Adds the forecast in the selected cell to the list of chosen forecasts
For *Index* = 2: Integer	cbChfClearList	5	Clears the list of chosen forecasts
For *Index* = 4: Boolean	True or False	n/a	True turns on vertical grid lines. False turns them off.
For *Index* = 5: Boolean	True or False	n/a	True turns on horizontal grid lines. False turns them off.
For *Index* = 6: Integer	cbTypReverseCumulCert	1	Displays the reverse cumulative certainties view of the chart
For *Index* = 6: Integer	cbTypCumulCert	3	Displays the cumulative certainties view of the chart
For *Index* = 6: Integer	cbTypCertCenteredOnMedian	4	Displays the chart with certainties centered on the median

Used With Specified Values of Index	Named Constant or Other Value	Index Value	Description
For *Index* = 7: Integer	cbValZeroBased	1	Uses zero-based value axes, normalizing the lowest forecast value to 0. This constant is included for compatibility with Crystal Ball 2000.*x* (5.*x*).
For *Index* = 7: Integer	cbValRelative	2	Uses relative value axes (the actual minimum and maximum values)
For *Index* = 7: Integer	cbValMinMaxDiv	3	Uses value axes by setting a minimum (*Value2*) and a maximum (*Value3*) so you can focus on a particular value range
For *Index* = 8: Integer	Any integer between 0 and 100 (inclusive)	n/a	Sets the first certainty band to display. Use this with *Value2* through *Value7*

Note: cbTypCertCenteredOnMean [2] is now obsolete.

Table 193, following, describes the *Value2* parameter.

Table 193 CB.TrendPrefsND Value2 Parameter Values — Optional, Variant

Used With Specified Parameters and Values	Value	Description
For *Index* = 7 and *Value1* = 3: Double	Any number	Defines the minimum value for the value axis
For *Index* = 8: Integer	Any integer between 0 and 100 (inclusive)	Sets a second certainty band to display. Use this with *Value1*.

Table 194, following, describes the *Value3* parameter.

Table 194 CB.TrendPrefsND Value3 Parameter Values — Optional, Variant

Used With Specified Parameters and Values	Value	Description
For *Index* = 7 and *Value1* = 3: Double	Any number	Defines the maximum value for the value axis
For *Index* = 8: Integer	Any integer between 0 and 100 (inclusive)	Sets a third certainty band to display. Use this with *Value1* and *Value2*.

When *Value1* is cbChfAdd, the active cell must contain a forecast; otherwise, CB.MacroResult returns cbErrNoForecastInCell. Crystal Ball adds the selected cell to the list of forecasts to appear in the trend chart.

To display a specific set of forecasts in a certain order on the trend chart, first clear the list of selected cells by calling this macro with *Value1* set to cbChfClearList. Then, for each forecast, use Range.Select to highlight the desired forecast cell and call this macro with *Value1* set to cbChfAdd.

You can set the percentage values of as many as seven certainty bands. Set *Index* to cbTrdCertaintyBands and supply the percentage values in *Value1* through *Value7*. They will appear in the trend chart in the order that you define them, so define them from smallest to largest or from largest to smallest.

See the *Oracle Crystal Ball User's Guide* for more information on how to customize the trend chart.

CB.TrendPrefsND Example

This example runs a simulation of 1000 trials, opens a trend chart, and selects a forecast cell. Then, it clears the list of forecasts included in the trend chart, adds the selected forecast to the trend chart, sets the value axis to have a minimum of 0 and a maximum of 1000, and includes certainty bands of 0%, 25%, 50%, 95%, and 100%.

```
CB.Simulation 1000
CB.OpenTrend
Range("B7").Select
CB.TrendPrefsND cbTrdChooseFore, cbChfClearList
CB.TrendPrefsND cbTrdChooseFore, cbChfAdd
CB.TrendPrefsND cbTrdValueAxis, cbValMinMaxDiv, 0, 1000
CB.TrendPrefsND cbTrdCertaintyBands, 0, 25, 50, 95, 100
```

CB.TrendPrefsND Related Call

Name	Description
CB.TrendPrefs	Opens the Trend Preferences dialog for the selected trend chart

CB.TwoDSimulation

This call lauches the 2D Simulation tool wizard. This tool estimates risk by running an outer loop to simulate uncertainty values, and then freezes the uncertainty values while it runs an inner loop (of the whole model) to simulate variability. This process repeats for some small number of outer simulations, providing a portrait of how a given forecast distribution varies due to the uncertainty.

Note: Before calling CB.TwoDSimulation, reset the simulation.

CB.TwoDSimulation Example

This example resets the Crystal Ball simulation and opens the 2D Simulation wizard.

```
CB.ResetND
CB.TwoDSimulation
```

CB.TwoDSimulation Related Calls

Name	Description
CB.TwoDSimulationND	Runs the 2D Simulation tool without displaying dialogs
CB.GetTwoDSimulation Option	Returns current settings for the 2D Simulation tool

CB.TwoDSimulationND

The 2D Simulation tool helps evaluate risk by estimating the uncertainty and variability for a given forecast using two-dimensional simulation. This call runs the 2D Simulation tool and displays results without using a dialog.

This call returns a variant.

Note: Before calling CB.TwoDSimulationND, reset the Crystal Ball simulation.

Table 195 CB.TwoDSimulationND Parameters

Parameter	Required?	VBA Data Type	Table
Index	Required	Integer	Table 196
Value1	Optional	Integer	Table 197
Value2	Optional	Variant	Table 198

Table 196 CB.TwoDSimulationND Index Parameter – Required, Integer

Named Constant Value	Index	Description
cbTwoDChooseTarget	1	Sets the target forecast for the 2D simulation operation (operates on the forecast in the currently selected cell; use [cell].Select first to select the cell)
cbTwoDChooseAssum	2	Works with *Value1* to choose assumptions for the uncertainty loop, the variability loop, or to clear the list (operates on the assumption in the currently selected cell; use [cell].Select first to select the cell)
cbTwoDOuterSimTrials	3	Used with *Value1* to specify the number of trials in the outer (uncertainty) simulation loop
cbTwoDInnerSimTrials	4	Used with *Value1* to specify the number of trials in the inner (variability) simulation loop
cbTwoDForeOption	5	Used with *Value1* to set forecast chart window display options
cbTwoDOutputOption	6	Used with *Value1* to set output report options
cbTwoDPercent	7	Used with *Value1* to specify percentile settings
cbTwoDMSDOption	8	Used with *Value1* to specify the maximum number of uncertainty simulations to display in the output report and overlay chart

Named Constant Value	Index	Description
cbTwoDRun	9	Runs the 2D Simulation tool

Table 197 CB.TwoDSimulationND Value1 Parameter – Optional, Integer

Related Value	Value or Named Constant	Index	Description
For *Index* = 2, cbTwoDChooseAssum: Integer	cbTwoDAddAssumUncert	1	Used with cbTwoDChooseAssum to add the selected assumption to the uncertainty simulation
For *Index* = 2, cbTwoDChooseAssum: Integer	cbTwoDAddAssumVar	2	Used with cbTwoDChooseAssum to add the selected assumption to the variability simulation
For *Index* = 2, cbTwoDChooseAssum: Integer	cbTwoDAssumClearList	3	Used with cbTwoDChooseAssum to return all unfrozen assumptions back to the uncertainty simulation
For *Index* = 3, cbTwoDOuterSimTrials	Positive whole number (integer)	n/a	Used with cbTwoDOuterSimTrials to specify the number of trials in the outer (uncertainty) simulation
For *Index* = 4, cbTwoDInnerSimTrials	Positive whole number (integer)	n/a	Used with cbTwoDInnerSimTrials to specify the number of trials in the inner (variability) simulation
For *Index* = 5, cbTwoDForeOption: Integer	cbTwoDShowDefineFore	1	Uses display settings for each forecast chart; the equivalent of the Show Forecasts As Defined dialog setting
For *Index* = 5, cbTwoDForeOption: Integer	cbTwoDShowTargetFore	2	Shows only the target forecast; the equivalent of the Show Only Target Forecast dialog settin
For *Index* = 5, cbTwo ForeOption: Integer	cbTwoDHideFore	3	Hides all forecasts; the equivalent of the Hide All Forecasts dialog setting
For *Index* = 6, cbTwoDOutputOption: Integer	cbTwoDShowForeStat	1	Used with cbTwoDOutputOption and a boolean *Value* parameter to specify whether to include forecast statistics in the output report; if followed by True, includes statistics in the report
For *Index* = 6, cbTwoDOutputOption: Integer	cbTwoDShowPercentiles	2	Used with cbTwoDOutputOption and a boolean *Value* parameter to specify whether to include percentiles in the output report; if followed by True, includes percentiles in the report
For *Index* = 6, cbTwoDOutputOption: Integer	cbTwoDShowCapMetrics	3	Used with cbTwoDOutputOption and a boolean *Value* parameter to specify whether to include capability metrics in the output report; if followed by True, includes capability metrics in the report
For *Index* = 7, cbTwoDPercent: Integer	cbTwoDPercentTenNinety	1	Specifies the 10th and 90th percentiles
For *Index* = 7, cbTwoDPercent: Integer	cbTwoDPercentFiveNinetyFive	2	Specifies the 5th and 95th percentiles
For *Index* = 7, cbTwoDPercent: Integer	cbTwoDPercentQuartiles	3	Specifies quartiles (25%-ile increments from 0 to 100, inclusive)

Related Value	Value or Named Constant	Index	Description
For *Index* = 7, cbTwoDPercent: Integer	cbTwoDPercentQuintiles	4	Specifies quintiles (20%-ile increments from 0 to 100, inclusive)
For *Index* = 7, cbTwoD Percent: Integer	cbTwoDPercentDeciles	5	Specifies deciles (10%-ile increments from 0 to 100, inclusive)
For *Index* = 7, cbTwoDPercent: Integer	cbTwoDPercentIcosatiles	6	Specifies icosatiles (5%-ile increments from 0 to 100, inclusive)
For *Index* = 7, cbTwoDPercent: Integer	cbTwoDPercentCustom	7	Defines custom percentiles, supplied by an optional *Value* entry (a one-dimensional cell range containing percentiles). If *Value* is omitted, the default percentiles are 2.5, 5, 50, 90, 95, 97.5.
For *Index* = 8, cbTwoDMSDOption: Integer	Integer from 2 to 250	n/a	Used with cbTwoDMSDOption to specify the maximum number of uncertainty simulations to display in the output report and overlay chart

Table 198 CB.TwoDSimulationND Value2 Parameter – Optional, Variant

Related Value	Value or Named Constant	VBA Data Type	Index	Description
For *Index* = 6, cbTwoDOutputOption: Integer	True or False	Boolean	n/a	Indicates whether to activate the specified *Value1* constant; if set to True, includes the specified items in the output report
For *Value1* = cbTwoDPercentCustom	A string containing a Microsoft Excel range that holds desired percentile values. The values must be entered into Microsoft Excel as numbers, not formatted as percentages; can be decimals.	String	n/a	Specifies custom percentile values to analyze

CB.TwoDSimulationND Example 1

This example resets the Crystal Ball simulation, selects a target forecast in cell A1, resets assumptions to the uncertainty list, adds cells to the variability list, specifies the number of trials for the uncertainty and variability simulations, specifies that only the target forecast chart should display, sets ouput options, sets the maximum number of uncertainty simulations to display in the ouput report and overlay chart to 75, and then runs the tool.

```
CB.ResetND

' Select target forecast
[A1].Select  '<- A1 must contain a forecast
CB.TwoDSimulationND cbTwoDChooseTarget

' Set all assumptions to the uncertainty list
CB.TwoDSimulationND cbTwoDChooseAssum, cbTwoDAssumClearList

' Set cells D3 and F9 to the Variability list
[D3].Select
```

```
CB.TwoDSimulationND cbTwoDChooseAssum, cbTwoDAddAssumVar
[F9].Select
CB.TwoDSimulationND cbTwoDChooseAssum, cbTwoDAddAssumVar

' Set the outer (uncertainty) simulation loop to 500 trials
CB.TwoDSimulationND cbTwoDOuterSimTrials, 500

' Set the inner (variability) simulation loop to 250 trials
CB.TwoDSimulationND cbTwoDInnerSimTrials, 250

' Set the forecast display to show only charts for the target forecast
CB.TwoDSimulationND cbTwoDForeOption, cbTwoDShowTargetFore

' Set the report options to include forecast statistics and custom
' percentiles
CB.TwoDSimulationND cbTwoDOutputOption, cbTwoDshowForeStat, True
CB.TwoDSimulationND cbTwoDOutputOption, cbTwoDshowPercentiles, True
CB.TwoDSimulationND cbTwoDPercent, cbTwoDPercentCustom, "F17:H17"

' Set the max number of uncertainty simulations to display in output to 75
CB.TwoDSimulationND cbTwoDMSDOption, 75

' Run the 2D Simulation tool
CB.Two2SimulationND cbTwoDRun
```

CB.TwoDSimulationND Example 2

This example runs the 2D Simulation tool using default preferences or settings from a previous run in the current Microsoft Excel session.

```
CB.ResetND

' Run the 2D Simulation tool
CB.TwoDSimulationND cbTwoDRun
```

CB.TwoDSimulationND Related Calls

Name	Description
CB.TwoDSimulation	Launches the 2D Simulation tool
CB.GetTwoDSimulationOption	Returns current settings for the 2D Simulation tool

CB.WorksheetProtection

Crystal Ball cannot run a simulation if a cell with Crystal Ball data is located in a write-protected worksheet. This subroutine is used to supply Crystal Ball with a worksheet password. After that, Crystal Ball uses this password to unlock the worksheet while running and lock it again afterwards.

The following commands will work after CB.WorksheetProtection is called:

- CB.Simulation

- CB.SingleStep
- CB.Reset
- CB.ResetND
- Microsoft Excel Save or SaveAs

> **Note:** Other calls that define or modify Crystal Ball data, such as CB.DefineAssum and CB.SetAssum, also require unprotected worksheets before they will run. However, these calls are not affected by CB.WorksheetProtection. Worksheets must be manually unprotected before data-modifying calls not listed previously will work.

Once the CB.WorksheetProtection command has run on each protected worksheet and the workbook saved, nothing else needs to be done for these commands to work.

Call CB.WorksheetProtection with the sheet you want to protect active. It only needs to be called through VBA (for example, from within the VBA Editor) one time for each sheet. Once the code has been run and the workbook saved, the code can be deleted and Crystal Ball will remember the password for future simulations.

It does not matter if the sheet is already protected when CB.WorksheetProtection is called as long as the password passed into the first parameter is consistent with the current worksheet password.

The password is encrypted so users cannot read it.

Table 199 CB.WorksheetProtection Parameters

Parameter	VBA Data Type	Value	Description
aPassword	String	Any alphanumeric string	Specifies the user-assigned password to register for that workbook
aTurnOn	Boolean	True or False	Specifies whether or not workbook protection is active. True activates protection and False removes it Use this option when you call this macro externally, such as from Visual Basic.

CB.WorksheetProtection Example 1

This example activates the target worksheet, passes the password "passwd," runs a simulation, and resets Crystal Ball .

```
Sub WorksheetProtectionExample1()
   ' The following call activates the target worksheet
   ThisWorkbook.Worksheets("Sheet3").Activate

   ' The following call passes current password to Crystal Ball
   CB.WorksheetProtection "passwd", True

   ' The following calls run and reset the simulation
   CB.Simulation 1000
   CB.ResetND
End Sub
```

Note: To apply protection to worksheets that do not have a password, use this:

```
CB.WorksheetProtection "", True
```

CB.WorksheetProtection Example 2

This example activates the target worksheet, unprotects the sheet with the current password, passes a new password, and protects the sheet.

```
Sub WorksheetProtectionExample2()

   ' The following call activates the target worksheet
   ThisWorkbook.Worksheets("Sheet3").Activate

   ' The following call unprotects sheet with current password
   CB.WorksheetProtection "passwd", False

   ' The following call passes a new password
   CB.WorksheetProtection "passwd2", False

   ' The following call protects the sheet
   CB.WorksheetProtection "passwd2", True

End Sub
```

CB.WorksheetProtection Example 3

This example activates the target worksheet, unprotects the sheet with the current password, clears the password by passing an empty string, and leaves the sheet unprotected.

```
Sub WorksheetProtectionExample3()
   ' The following call activates the target worksheet
   ThisWorkbook.Worksheets("Sheet3").Activate

   ' The following call unprotects sheet with current password   CB.WorksheetProtection
"passwd2", False


   ' following call clears the password
   CB.WorksheetProtection "", False
End Sub
```

4 Changes from Previous Versions

In This Chapter

Introduction

This appendix lists differences between the 2000.5 (5.5) version of the Crystal Ball Developer Kit and this version. Most of these differences also apply to other 2000.*x* (5.*x*) versions.

You can learn about new calls now available for use in future code and can use the lists of omitted and changed calls to troubleshoot conversion problems in existing code.

Added in Crystal Ball 11.1.*x*

The following new calls were added to version 11.1.1.0.00 and 11.1.2.0.00 of the Crystal Ball Developer Kit. They support new Crystal Ball tools and parameter locking features in distribution fitting.

Note: Calls added in version 11.1.2.0.00 are marked with an asterisk (*)

CB.BatchFit

CB.BatchFitND

CB.Bootstrap

CB.BootstrapND

CB.DataAnalysis

CB.DataAnalysisND

CB.DecisionTable

CB.DecisionTableND

CB.DefineAssumFromForeND *

CB.LockFitParm

CB.GetBatchFitOption

CB.GetBootstrapOption

CB.GetDataAnalysisOption

CB.GetDecisionTableOption

CB.GetLockFitParm

CB.GetScenarioAnalysisOption *

CB.GetTwoDSimulationOption

CB.ScenarioAnalysis *

CB.ScenarioAnalysisND *

CB.TwoDSimulation

CB.TwoDSimulationND

Added in Crystal Ball 7.1 - 7.3.x

The following new calls have been added to previous versions of the Crystal Ball Developer Kit since Crystal Ball 2000.5 (5.5). An asterisk (*) indicates that the call has changed from the previous version in some way. Check "Changes to Existing Calls" on page 237 for details.

CB.AlertOnArgumentError

CB.AlertOnMacroResultError

CB.AlertOnObsolete

CB.AutoDownshift

CB.CBLoaded

CB.CheckDataND

CB.CloseAllCharts

CB.CloseChart

CB.CloseSensitiv*

CB.CloseTrend*

CB.CopyScatter

CB.CopySensitiv

CB.CopyTrend

CB.CreateChart

CB.DefineAltParms*

CB.DeleteChart

CB.EnumChart

CB.EnumCorrelation

CB.Fit*

CB.Freeze

CB.FreezeND

CB.GetCBAutoLoad

CB.GetExcel2007ForegroundMode

CB.GetFitParm*

CB.GetForeData

CB.GetVersion*

CB.GetWorksheetVersion

CB.MacroResultDetail

CB.OpenChart

CB.OpenSelection

CB.OpenSensitiv*

CB.OpenTrend*

CB.RestoreResults

CB.RestoreResultsND

CB.RuntimeMode

CB.SaveResults

CB.SaveResultsND

CB.SelectChart

CB.SensPrefs, CB.SensPrefsND*

CB.SetCBWorkbookPriority

CB.SetFitRange

CB.TrendPrefs, CB.TrendPrefsND*

CB.ScatterPrefs

CB.ScatterPrefsND

CB.SetCBAutoLoad

CB.SetExcel2007ForegroundMode

CB.Shutdown

CB.Startup

CB.WorksheetProtection

Not Included in This Version

The following calls available in Crystal Ball 2000.*x* (5.*x)* or earlier versions are not currently available in this version of the Crystal Ball Developer Kit:

CB.Auto_Open (see CB.Startup)

CB.CloseCB (see CB.Shutdown)

CB.CloseOverlay

CB.CopyCertainty

CB.CopyFore

CB.CopyOverlay

CB.CopyRangeMax

CB.CopyRangeMin

CB.ForeWindows

CB.FreezeAssum (replaced by CB.Freeze)

CB.FreezeAssumND (replaced by CB.FreezeND)

CB.Help

CB.OpenOverlay

CB.OverlayPrefs

CB.OverlayPrefsND

CB.RangePrefs

CB.RangePrefsND

CB.ResampleFore

CB.RestoreRun (replaced by CB.RestoreResults)

CB.RestoreRunND (replaced by CB.RestoreResultsND)

CB.SaveRun (replaced by CB.SaveResults)

CB.SaveRunND (replaced by CB.SaveResultsND)

CB.SelectSome

CB.SetForeTrial

CB.SetView

CB.StatsPrefsND

Regarding CB.StatsPrefsND, the following describes why it was omitted from this version of the Crystal Ball Developer Kit: the constants cbStaBaseRange and cbRngDisplay are obsolete since their functionality is replaced by the filtering constants in CB.SetFore, the constant cbStaCenterOn is obsolete because ranges are now always centered on Median, and cbStaReversePercentile is obsolete because it is now handled in CB.RunPrefsND.

Note: To support the changes from CB.RestoreRun and CB.RestoreRunND to CB.RestoreResults and CB.RestoreResultsND, the related constant returned from CB.MacroResult has been changed from cbErrRestoreRun to cbErrRestoreResult. The returned value is unchanged.

The following spreadsheet functions are not included in this version of the Crystal Ball Developer Kit:

CB.ExtremeValue

CB.ExtremeValue2

CB.GetCorrelationFN

CB.GetDecVarFN

CB.GetForeFN

CB.GetVersionFN

Changes to Existing Calls

This section lists macro calls that have changed since the 2000.5 (5.5) version of the Crystal Ball Developer Kit and describes the changes.

In the following updates, constant index numbers are indicated in square brackets. For example, cbCelDistMedian [5] means constant cbCelDistMedian with index number 5.

When constants are labeled "deprecated," they are currently accepted for compatibility but arecome obsolete in a future version and should not be used in new code.

CB.AssumPrefsND

- Documentation errors corrected: the first parameter is renamed PrefItem and the index number for constant cbAssumShowMean is changed from 2 to 3 to match the code.
- The constant cbAssumShowCoordinates is removed (obsolete in Crystal Ball 7.*x* and later, including 11.*x*).

CB.AutoDownshift

- Starting with Crystal Ball version 7.2, this call changes from a subroutine to a function and returns the boolean setting of the TurnOn parameter, True (if CB.AutoDownshift is set to automatically "downshift" to Normal speed) or False (if it is not set to downshift).

CB.BatchFitND

- Starting with Crystal Ball version 11.1.2.1.000, two new constants are added to Index:
  - cbBftOutputOrientation [30] with Value1 constants cbBftOutputRows [1] and cbBftOutputColumns [2]
  - cbBftOutputSheetName [31], which sets the name of the worksheet containing primary fit results (assumptions) as the string given for Value1

CB.CellPrefsND

- Index2 parameter — new constants added:
  - cbCelDistMedian [5]
  - cbCelRangeMin [6]
- Value parameter — constant cbCelPattern [1] has 0-17 values instead of 0-3; additional values are added for the new Index2 constants with indices 5-7.
- Level parameter (optional) added to support the new Apply To dialog control with new constants:
  - cbCelGlobal [0]
  - cbCelWorkbook [1]
  - cbCelWorksheet [2]

CB.ChartPrefsND

- Index parameter — constant removed (obsolete): cbChtRedrawFrequency [4].

CB.CheckData

- Should now be called each time data is deleted or added as well as each time a new enumeration is started.

CB.ClearData

- If the selected range contains more than one type of data, a dialog appears and prompts for the type of data to clear.
- Simulations must now be reset before CB.ClearData is called.

CB.ClearDataND

- Simulations must now be reset before CB.ClearDataND is called.

CB.CloseFore

- Can now be called while a simulation is running.

CB.CloseSensitiv

- A chart can now be explicitly selected (via CB.EnumChart and CB.SelectChart) before it can be closed.

CB.CloseTrend

- A chart can now be explicitly selected (via CB.EnumChart and CB.SelectChart) before it can be closed.

CB.CopyData and CB.CopyDataND

- Data is no longer copied to Crystal Ball's internal clipboard. Instead, these calls record a range of data to be pasted. All Crystal Ball data within that range is pasted when CB.PasteData runs. For this reason, CB.PasteData should be called immediately after CB.CopyData or CB.CopyDataND, before any data is changed within the copied range.
- For CB.CopyDataND, the documentation is corrected to match the code: parameter CBData is renamed DataType.

CB.CorrelateND

- A new optional parameter, SecondAssum, is added to specify a second cell to correlate with the cell specified by the first parameter. The cell specified by SecondAssum can be on another worksheet in the same workbook.

CB.CreateRpt

- Can now be called while a simulation is running.

CB.CreateRptND

- Can now be called while a simulation is running.
- Is substantially changed to support the revised Create Report command in Crystal Ball 7.*x* and later (including 11.*x*):

- The Index constant cbRptForeFrequencies [8] is removed (obsolete).
- Index constant cbRptDecVar [15] is deprecated. cbRptChooseDecVar [18] should be used instead.
- Index constant RptExistingSheet [16] is changed so that True creates the report on a new sheet instead of the existing sheet of the current workbook while False creates the report in a new workbook.
- New Index constants with index values of 17-26, 28-30, and 32-37 are added as described in Table 31.
- Additional constants for parameters Value1 and Value2 are added to support the new Index constants. They are listed in Table 36 through Table 40.
- Unlike in pre-7.0 versions, the following statement needs to be added at the beginning of the Create Report code to create a custom report:

  CB.CreateRptND cbRptDefinedType, cbRptCustom

  Otherwise, the type of report currently selected in the Create Report dialog (by default, the Full report) is created.

- For Index constants supported in Crystal Ball 2000.5 (5.5) the following changes appear:
  - cbRptForePercentiles — two new constants are added for Value2 (described in Table 32): cbPctSet3 [7], cbPctSet4 [8].
  - cbRptChooseAssum — a new constant is added for Value1: cbChaOpen [5], described in Table 34.
  - cbRptChartType — the constant cbChtBW [0] is now obsolete and has been removed.
- To support the rewritten version of OptQuest added in version 11.1.1.0.00:
  - The following new Index constants are added: cbRptOptimizerSummary [39], cbRptOptimizerChart [40], cbRptOptimizerBestSolution [41], cbRptOptimizerConstraints [42], cbRptOptimizerDecVars [43].
  - A new report section type is available as Value2 for cbRptSection: cbRptSectOptimizerResults [9].
  - A new Value2 constant is available for cbRptDefinedType: cbRptOptimizer [7].
  - A ninth integer is available for cbRptSectionOrder; the cbRptSectOptimizerResults section is assigned number 9.
- To support the rewritten version of Predictor added in version 11.1.3.0.00:
  - The following new Index constants are added: cbRptPredictorChart [45], cbRptPredictorStatistics [46], cbRptPredictorAutocorrelations [47], cbRptPredictorForecast [48], cbRptPredictorConfidenceIntervals [49], and cbRptPredictorMethods [50].
  - A new report section type is available as Value2 for cbRptSection: cbRptSectPredictorSeries [10].
  - New Value1 constants are available for cbRptPredictorMethods: cbPredictorMethodsNone [1], cbPredictorMethodsAll [2], cbPredictorMethods1Best [3], cbPredictorMethods2Best [4], cbPredictorMethods3Best [5].

- A tenth integer is available for cbRptSectionOrder; the cbRptSectPredictorSeries section is assigned number 10.

CB.DefineAltParms

- An optional fourth parameter is added to support the version of the beta distribution used in versions later than Crystal Ball 2000.5 (5.5).
- cbParmMidpoint [-20] and cbParmDegFreedom [-21] are added to support the Student's *t* distribution, added in Crystal Ball 7.0.
- Log and geometric mean and standard deviation alternate parameter sets can now be used with lognormal distributions.

  Note: Because there are three alternate parameter sets (in addition to percentiles) for the lognormal distribution, the cbParmRegular named constant cannot be passed to CB.DefineAltParms when setting up an alternate parameter set for the lognormal distribution. You must explicitly set the parameter set with, for example, cbParmLogMean and cbParmLogStDev.
- cbParmMode [15] is included to support the Extreme Value distribution for backward compatibility with Crystal Ball 2000.x (5.x). In later versions of Crystal Ball, that distribution is replaced with the Maximum Extreme and Minimum Extreme distributions. cbParmMode is deprecated in this version of Crystal Ball and are obsolete in a future version.
- The implementation of DefineAltParms in Crystal Ball 7.*x* and later versions (including 11.*x*) is different from Crystal Ball 2000.5 (5.5) in that a different integer is returned for each call even if the parameters are the same. This number is simply a pointer to the alternate parameter set. Code written in Crystal Ball 2000.5 (5.5) should run the same in Crystal Ball 7.2 or later (including 11.*x*).

CB.DefineAssum

- If a simulation was run earlier, it must be reset before this call runs.

CB.DefineAssumND

- If a simulation was run earlier, it must be reset before this call runs.
- The optional Dynamic parameter is deprecated in this release since all cell references are now dynamic. It should not be used in new code.
- An optional fourth parameter, Parm4, is added to support the DistType constant cbDfaBeta, which now has four parameters.
- The following new constants are provided for DistType to support new distributions added in versions of Crystal Ball beginning with Crystal Ball 7.0 :
  - cbDfaHypergeometricSuccess [17]
  - cbDfaMinExtreme [18]

- cbDfaMaxExtreme [19]
- cbDfaStudentsT [20]
- cbDfaYesNo [21]
- cbDfaDiscreteUniform [22]
- cbDfaBetaPert [23] (added in Crystal Ball 7.3)

- cbDfaHypergeometric [10] and cbDfaExtremeValue [15] are deprecated in this release. Use cbDfaHypergeometricSuccess [17] instead of cbDFAHypergeometric [10] and use cbDfaMinExtreme [18] or cbDfaMaxExtreme [19] instead of cbDfaExtremeValue [15]. Table 53 lists the values for Parm1, Parm2, Parm3, and Parm4 for the new DistType constants.

CB.DefineDecVar

- If a simulation was run earlier, it must be reset before CB.DefineDecVar is called.

CB.DefineDecVarND

- If a simulation was run earlier, it must be reset before CB.DefineDecVarND is called.
- Beginning in Crystal Ball 11.1.1.0.00, there are three new decision variable types: Binary, Category, and Custom (described in Table 56).
- To support these new types, a new optional parameter, decVarType, is added with the following new constants: cbDecVarTypeContinuous [0], cbDecVarTypeDiscrete [1], cbDecVarTypeBinary [2], cbDecVarTypeCategory [3], and cbDecVarTypeCustom [4].
- To support decVarType equal to cbDecVarTypeCustom, a new optional parameter, customData, is added.

CB.DefineForeND

- All parameters are now optional.
- All parameter data types are now Variant.
- The optional WindowSize parameter is obsolete and has been removed.
- New optional parameters have been added to support the process capability features: LSL, USL, and Target.

CB.EnumAssum

- Starting with Crystal Ball l 7.2, this function now returns absolute cell references.

CB.EnumDecVar

- Starting with Crystal Ball 7.2, this function now returns absolute cell references.

CB.EnumFore

- Starting with Crystal Ball 7.2, this function now returns absolute cell references.

CB.ExtractDataND

- The Extract Data feature and dialogs have changed substantially since Crystal Ball 2000.5 (5.5). CB.ExtractDataND has changed to support the new functionality:
  - The Value constant cbDatForeValues [1] is deprecated in this release; cbDatValues [1] should be used instead.
  - The description of Value constant cbChfChosen [3] corrects a documentation error. Use this constant after cbChfAdd, not cbChaAdd.
  - The Value constant cbDatCumulative [5] is not included in this release.
  - A new Value constant cbDatCapMetrics [7] is added for Index constant cbExtDataType [2].
  - Value constants cbPctSet3 [7] and cbPctSet4 [8] are added to support new product functionality.
  - For Index constant cbExtExistingSheet [5], Value constants can be cbOKExistingSheet or cbOKNewSheet as well as True or False, respectively.
  - Additional Index constants are added: cbExtChooseAsm [6], cbExtChartBins [7], and cbExtIncludeCellLoc [8] (Crystal Ball 7.2), plus cbExtIncludeLabels [9], cbExtAutoFormat [10], cbExtSheetName [11], cbExtStartCell [12], and cbExtStartSheet [13] (Crystal Ball 7.2.1).
  - Starting with Crystal Ball 11.1.2.0.00, a new constant, cbExtChartBinsUseEntireRange [28], is added to the Index parameter to indicate whether to display all values including extreme values that are not displayed in the chart or just the values displayed in the chart.
- For more information on these constants and other changes, see Table 67.
- To support the rewritten version of OptQuest added in version 11.1.1.0.00:
  - The following Index constants are added: cbExtDataOptimizerType [14], cbExtChooseDecVar [15].
  - The following Value constants are added for cbExtDataOptimizerType: cbDatOptimizerSolutions [1], cbDatOptimizerStatistics [2].
  - The following Value constants are added for cbExtChooseDecVar: cbChooseDecVarAll [1], cbChooseDecVarChosen [2], cbChooseDecVarAdd [3], cbChooseDecVarClearList [4].

- To support the rewritten version of Predictor added in version 11.1.1.3.00, the following Index constants are added: cbExtPredictorResultsSheetName [17], cbExtPredictorMethodsSheetName [18], cbExtPredictorResultsHistorical [19], cbExtPredictorResultsFitted [20], cbExtPredictorResultsForecast [21], cbExtPredictorResultsConfidence [22], cbExtPredictorResultsResiduals [23], cbExtPredictorMethodsErrors [24], cbExtPredictorMethodsStatistics [25], cbExtPredictorMethodsParameters [26], and cbExtPredictorMethodsRanking [27].

CB.Fit

- CB.Fit must now be used with CB.SetFitRange to select the range of data to fit to a distribution.
- Constants have been added to support distributions added in Crystal Ball 7: cbDfaMinExtreme [18], cbDfaMaxExtreme [19], cbDfaStudentsT [20], and cbDfaBetaPert [23] (added in Crystal Ball 7.3).
- Constant cbDfaExtremeValue [15] is deprecated. It is included for Crystal Ball 2000.5 (5.5) compatibility and is replaced by cbDfaMinExtreme and cbDfaMaxExtreme in later versions of Crystal Ball .
- Criterion constant cbFitChiSquareP [2] has been added back in Crystal Ball 11.1.1.0.00 to show the Chi-square *p*-value.
- A new optional boolean parameter, pValue, is added to display *p*-values instead of statistics when set to True.
- Starting with Crystal Ball 11.1.1.0.00, discrete distributions can be used for fitting and Crystal Ball will also select distributions and a ranking method automatically when requested. The following new Dist parameter constants support new distribution fitting options: cbDfaAutoSelectDist[50], cbDfaAllContinuous [51], cbDfaAllDiscrete [52], and cbDfaAllDistributions [53]. A new Criterion parameter constant, cbDfaAutoSelectRanking Method [5], has been added to support automatic selection of the ranking method.

CB.Freeze and CB.FreezeND

- These calls replace CB.FreezeAssum and CB.FreezeAssumND. This change reflects the new functionality in Crystal Ball versions later than 2000.5 (5.5) that can freeze decision variable and forecast cells as well as assumptions.

CB.GetAssum

- To correct a documentation error, the parameter name AssumRef is changed to AssumReference to match the code. The data type is now Variant.

CB.GetAssumPercent

- The data type of the Percent parameter is now Double.

CB.GetBatchFitOption

- Starting with Crystal Ball version 11.1.2.1.000, two new constants are added to Index:
  - cbBftOutputOrientation [30], which returns cbBftOutputRows [1] or cbBftOutputColumns [2]
  - cbBftOutputSheetName [31], which returns the name of the worksheet containing primary fit results (assumptions) as a string

CB.GetCertainty

- The first parameter is now named ForeReference with a data type of Variant.

CB.GetCorrelation

- To correct a documentation error, the first parameter is now named AssumReference with a data type of Variant.
- The Row parameter data type is now Long.

CB.GetDecVar

- To correct a documentation error, the first parameter is now named DecVarReference with a data type of Variant.
- A new constant is added for the Index parameter: cbDecFrozen [6].
- Starting with Crystal Ball 11.1.1.0.00, the cbDecType [4] constant now returns five types: Continuous [0], Discrete [1], Binary [2], Category [3], Custom [4].
- Also, a new constant, cbDecCustomData [7], is added to the Index parameter for use when cbDecType [4] indicates a Custom [4] type.
- Starting with Crystal Ball 11.1.2.0.00, a new constant, cbDecCustomDataCount [8], is added to the Index parameter for use when cbDecType [4] indicates a Custom [4] type.

CB.GetFitParm

- A fourth Index value is added to support the 7.*x* and 11.*x* beta distribution.
- The parameters for beta have changed since 2000.*x* (5.*x*). A fourth Index value is added to support the 7.*x* and 11.*x* beta distribution.
- The extreme value constant is deprecated. It is included for Crystal Ball 2000.5 (5.5) compatibility and is replaced by cbDfaMinExtreme and cbDfaMaxExtreme in later versions of Crystal Ball l.
- A new distribution type, betaPERT (cbDfaBetaPert), is added in Crystal Ball 7.3.

CB.GetFore

- To correct a documentation error, the first parameter is now named ForeReference with a data type of Variant.
- The Index parameter has these constant changes:
  - cbForWindowSize [3] is obsolete and has been removed.
  - cbPrecisionPerc [12] is renamed to cbForPrecisionPerc to correct a documentation error.
  - cbPrecisionPercValue [13] is renamed to cbForPrecisionPercValue to correct a documentation error.
  - cbForFilterGlobal [18] was obsolete in the Crystal Ball 7.1 version of the Developer Kit. It is now valid and has been added back.
  - cbForFrozen [19], cbForLSL [20], cbForUSL [21], and cbForTarget [22] are added.

CB.GetForeStat

- New constants with index numbers 50 through 64 have been added to support Crystal Ball's process capability features.
- In version 11.1.1.3.00, a new constant cbFstBaseCase [14] was added to return the base case (initial value) of a forecast cell.

CB.GetRunPrefs

- The additional parameters listed in the Developer Kit documentation for Crystal Ball 2000.5 (5.5), cbRunStdError and cbRunBurstAmount, are obsolete in this version of the Crystal Ball Developer Kit.
- Starting with Crystal Ball 11.1.2.1.000, cbRunAlertLevel [62] constants are added to specify the level of alert messages to show and to reset message display preferences and can be returned by CB.GetRunPrefs.

CB.GetVersion, CB.GetWorksheetVersion

- Starting with version 11.1.1.0.00, two optional parameters, ValueType and Index, return the full or partial version ID or build number. Full IDs or numbers are returned as strings and partial IDs or numbers are returned as an integer.

CB.IsCBObject

- The parameter SheetObj changes to AnObject; its data type changes from Object to Variant.

CB.OpenSensitiv

- A chart can now be explicitly selected (via CB.EnumChart and CB.SelectChart) before it can be opened.

CB.OpenTrend

- A chart can now be explicitly selected (via CB.EnumChart and CB.SelectChart) before it can be opened.

CB.PasteData

- This subroutine no longer pastes from Crystal Ball's internal clipboard. Instead, it pastes all Crystal Ball data currently existing within the range specified by CB.CopyData or CB.CopyDataND, to the extent that appropriate value or formula cells exist within the range selected for pasting. For best results, use this subroutine immediately after CB.CopyData or CB.CopyDataND.

CB.RunPrefsND

- The following Index constants used in Crystal Ball 2000.5 have changed as indicated:
  - cbRunBurstMode [5], obsolete and removed from the code
  - cbRunResetToOriginal [6], obsolete and removed from the code
  - cbRunSensitivAnalysis [9], obsolete and removed from the code
  - cbRunRetainUnsortedTrials [11], obsolete and removed from the code
  - cbRunPrecisionWithin [45], obsolete and removed from the code
  - cbRunLeaveOpenOnReset [50], new in Crystal Ball 7.0
  - cbRunMacroBeforeSimul [21], obsolete and removed from the code; replaced by cbRunUserMacros [51]
  - cbRunMacroBeforeRecalc [22], obsolete and removed from the code; replaced by cbRunUserMacros [51]
  - cbRunMacroAfterRecalc [23], obsolete and removed from the code; replaced by cbRunUserMacros [51]
  - cbRunMacroAfterSimul [24], obsolete and removed from the code; replaced by cbRunUserMacros [51]
  - cbRunMinimizeSheets [27], changed to work as described in the Value parameter section of Table 152
  - In Crystal Ball 2000.5 (5.5), values for cbRunMode [29] were booleans; in later versions of Crystal Ball, the values are integers. Models from Crystal Ball 2000.*x* that use this constant will not run successfully in later versions of Crystal Ball until the constant values are edited.

- cbRunParadiseServer [30], obsolete and removed from the code
- cbRunBurstAmount [33], obsolete and removed from the code
- New constants added include cbRunPrecisionWithin [45], cbRunPrecisionConfidence [46], cbRunFormatPercentiles [48], cbRunSaveAssumptionValues [49], cbRunUserMacros [51], cbRunCapMetrics [52], cbRunCapMetricsShort [53], cbRunCapMetricsShift [54], cbRunCapMetricsCalcType [55], cbRunCapMetricsNormalityThresh [56], cbRunCapMetricsNormalCalcOptions [57].

- cbRunPrecisionMean [41], used in versions before Crystal Ball 2000.5 (5.5), is now obsolete.
- The equivalent Value1 and Value 2 constants are removed or added as appropriate.
- Starting with Crystal Ball 11.1.2.1.000, cbRunAlertLevel [62] and related constants are added to specify the level of alert messages to show and to reset message display preferences.

CB.SensPrefsND

- cbSenIncludeAssum [3] is now obsolete; cbSenChooseAssum [8] is used in its place.
- cbSenIncludeOtherFore [4] is now obsolete.
- cbSenTitle [7] has been added.

CB.SetAssum

- Data type returned changes from boolean to Variant.
- The first parameter name is changed from AssumRef to AssumReference to correct a documentation error.
- The data type of the first parameter is changed from Range to Variant.

CB.SetDecVar

- Data type returned changes from boolean to Variant.
- The first parameter name is changed from DecVarRef to DecVarReference to correct a documentation error.
- The data type of the first parameter is changed from Range to Variant.
- There is a new Index constant, cbDecFrozen [6].
- Starting with Crystal Ball 11.1.1.0.00, two new Index constants are added:
  - cbDecType [4] with Value constants cbDecVarTypeContinuous [0], cbDecVarTypeDiscrete [1], cbDecVarTypeBinary [2], cbDecVarTypeCategory [3], and cbDecVarTypeCustom [4].
  - cbDecCustomData [7], which supplies a data string for Value = cbDecVarTypeCustom [4].

CB.SetFore

- Data type returned changes from boolean to Variant.
- The first parameter name is changed from ForeRef to ForeReference to correct a documentation error.
- Index constant cbForWindowSize [3] is obsolete and is removed.
- Index constant cbForFilterGlobal [18] was obsolete in the Crystal Ball 7.1 version of the Developer Kit. It is now valid and has been added back.
- Index constants cbForFrozen [19], cbForLSL [20], cbForUSL [21], and cbForTarget [22] are new.
- Value constants are removed or added accordingly.

CB.SimResult

- Was previously documented incorrectly in the Crystal Ball 2000.5 Developer Kit as CB.SimResultND.

CB.Simulation

- Optional parameters HideForecasts, MinimizeHost, MinimizeAllSheets, and StatusString have changed their data types to Variant.
- ShowControlPanel is an optional boolean parameter added for Crystal Ball 7.0.
- Starting with Crystal Ball 11.1.1.0.00, the optional Asynchronous parameter is obsolete. For placeholder purposes, this parameter remains in the code and has been renamed to Obsolete_Asynchronous. If any other optional parameters are used, a double set of commas (,,) should appear between Iterations and the other optional parameters to show that there is another parameter with no value between them.

CB.StartMultiSimul

Has three optional parameters: Rescan, SaveAssumValues, and MultiAutoDownshift. The last two were added in Crystal Ball 7.2.1.

CB.TrendPrefsND

- cbTrdForeTitleAxis [3] is now obsolete.
- cbTypCertCenteredOnMean [2] is now obsolete.
- cbTypCertCenteredOnMedian [4] is added.
- Certainty band values for Index constant cbTrdCertaintyBands no longer need to be divisible by 5.

Other Changes

These are general changes that do not apply to a particular call.

Charts Module

The Crystal Ball 2000.5 (5.5) Developer Kit included a Charts module in the cb.xla code file. Because of the way charts are created in later versions of Crystal Ball, that module is no longer included in the Developer Kit.

5 Crystal Ball Runtime

In This Chapter

Introduction

Application developers can use the current version of the Crystal Ball Developer Kit to create custom applications for others who are not skilled users of Crystal Ball.

This feature is called Crystal Ball Runtime. This appendix describes Crystal Ball Runtime and explains how to install and use it.

About Crystal Ball Runtime

When your custom application calls CB.RuntimeMode ("CB.RuntimeMode" on page 191), the Crystal Ball menus and toolbar (or ribbon in Microsoft Excel 2007) are hidden. You can give users the ability to run, reset, and single-step through simulations by creating your own user interface controls with Visual Basic or VBA that call appropriate Crystal Ball Developer Kit subroutines and functions. You can also use the Crystal Ball Developer Kit macros to create controls that extract data, create charts and reports, and more.

When simulations run in Runtime mode, the Crystal Ball Control Panel is switched off by default. However, you can use Visual Basic or VBA to switch it on again using appropriate calls to CB.Simulation or CB.RunPrefsND. Even if the Control Panel dialog is displayed, though, the Run, Analyze, and Help menus are hidden.

Users can access Predictor if you use calls from the Predictor Developer Kit to access Predictor features. See Appendix B, "Using the Predictor Developer Kit."

Runtime Example Workbook

Crystal Ball includes Runtime Example.xls, a very basic application written in VBA using the Crystal Ball Developer Kit to show how UI controls — in this case, buttons — can be created to

operate a Crystal Ball Runtime application. You can open this example in the Examples folder under the folder where Crystal Ball is installed.

Delivering Crystal Ball Runtime Applications

An appropriate Crystal Ball license must be purchased for each person who will use your custom application.

Typically, Crystal Ball Runtime applications are delivered as .xls or .xla files. These are then combined (for example, zipped) with the Crystal Ball setup file for the version of Crystal Ball used to develop the Runtime application for delivery to application users. You also need to include an installation script to run the setup file and a licensing script to pass the appropriate license serial number to the Crystal Ball License Manager.

Alternately, you can have application users unzip the application and Crystal Ball setup file, double-click the setup file to launch it and install Crystal Ball , and then run the Crystal Ball License Manager to license Crystal Ball using the license serial number provided for each user. Instructions are the same as those given in the current *Crystal Ball Installation and Licensing Guide*.

Note: Users of Crystal Ball Runtime applications must meet the usual Crystal Ball system requirements. For more information, see "System Requirements and .NET" on page 253.

Installation Script

The application setup script should include this command:

```
setup.exe /s /v"/qn /norestart /lv c:\install.log"
```

where

- `setup.exe` is the executable for the appropriateCrystal Ball installer. If you downloaded your version of Crystal Ball from the Web, the filename can include the version number, such as cb111.exe, and that name should appear in the script.
- `/s` = silent mode
- `/v` = pass arguments to msiexec
- `/qn` = do not display the UI
- `/norestart` = do not restart
- `/lv` <*logfile*> = verbose output, which writes the file c:\install.log

To add a progress bar, insert `/passive` as an argument between `/qn` and `/norestart`.

Note: Using the silent install on Windows Vista is similar, however you have to right-click and choose Run As Administrator when opening the Vista Command Prompt.

Because the arguments passed in with the `/v` switch are arguments to the Windows Installer (.msi), you can use the following to perform a silent install with the Windows Installer (if you have extracted Crystal Ball .msi from setup.exe):

```
msiexec /i "Crystal Ball .msi" /qn /norestart /lv c:\install.log
```

System Requirements and .NET

For current system requirements for running Crystal Ball see the current *Oracle Crystal Ball Installation and Licensing Guide*.

The installation script described in the previous section will only run if all system requirements are met, including the availability of Microsoft .NET Framework 2.0, 3.0, or 3.5.

If users have another version of the Microsoft .NET Framework, they might get an error when starting Crystal Ball Runtime. You can either instruct them to run the Crystal Ball Application Manager and check the Microsoft .NET Framework 2.0, 3.0, or 3.5 checkbox, or you can create (or update) an excel.exe.config file to ensure Crystal Ball runs against Microsoft .NET Framework 2.0, 3.0, or 3.5.

To create or update excel.exe.config, run the following script after the installation script:

```
AddInManager /ConfigExcelForDotNet
```

This script should be adjusted to run the Crystal Ball AddInManager from the Bin folder beneath the main Crystal Ball installation folder, by default `C:\Program Files\Oracle\Crystal Ball\Bin`. The resulting file, excel.exe.config, appears in `C:\Program Files\Microsoft Office\Office #`, where `Office #` indicates the default version of Microsoft Office (including Microsoft Excel) that Crystal Ball opens when it starts.

If excel.exe.config already exists, the information relevant to Crystal Ball is appended to it or updated, as appropriate. The new information in excel.exe.config indicates that Crystal Ball should use Microsoft .NET Framework 2.0, 3.0 or 3.5.

Licensing Script

A major part of Crystal Ball Runtime licensing is entering a license serial number so those who receive Crystal Ball Runtime applications are able to run them on their computers. The serial number is supplied by the Crystal Ball team for distribution through direct installation and licensing or through scripts and allows a certain number of activations.

Users of your applications can install and license Crystal Ball directly, following instructions in the current *Oracle Crystal Ball Installation and Licensing Guide*. Alternatively, you can use a script for licensing, also described in the *Oracle Crystal Ball Installation and Licensing Guide*.

Note: C:\Users\<username>\Documents\output.txt

Startup Scripts

Ideally, users shouldn't need to manually start Crystal Ball. You can use scripting to make sure that Crystal Ball automatically loads with Microsoft Excel, or you can start the Crystal Ball launcher file (CBLauncher.exe), installed with Crystal Ball, to make sure Crystal Ballstarts, as follows:

- To ensure that Crystal Ballstarts automatically when Microsoft Excel starts, set the following registry key to a value of 3:

```
HKEY_CURRENT_USER\Software\Microsoft\Office\Excel\Addins\SecureCBAddin.Connect
\LoadBehavior
```

  This is the same as checking the checkbox to automatically launch Crystal Ballwhen Microsoft Excel starts in the Crystal BallApplication Manager.

  Note: Be sure to set the registry key value back to 2 when the application closes.

  OR,

- To confirm that Crystal Ballis running before you run any Crystal BallDeveloper Kit code in the custom application, call CBLauncher.exe as follows:

```
C:\Program Files\Oracle\Crystal Ball\Bin\CBLauncher.exe
```

  where the installation folder part of this call should be modified if CB is installed on a non-English machine or to a non-default installation folder.

  OR,

- You can use new Crystal Ball Developer Kit calls to open and close Crystal Ball and determine load status: CB.CBLoaded, CB.GetCBAutoLoad, CB.SetCBAutoLoad, CB.Shutdown, and CB.Startup. For more information about these calls, see Chapter 3, "Crystal Ball Macro Calls."

Installing Crystal Ball Runtime

➤ To install your custom applications and Crystal Ball with a Runtime license, instruct users to:

1 **Set Microsoft Excel macro security to Medium (recommended) or Low using the following command sequence in Microsoft Excel: Tools, then Options, then Security, and then Macro Security.**

  In Microsoft Excel 2007, click the Office button and click Excel Options. Then, check Show Developer Tab In The Ribbon and click OK. On the Developer tab, choose Macro Security. Then, check "Disable all macros with notification" (recommended) or "Enable all macros (not recommended; potentially dangerous code can run)."

  Note: If you have attached a digital signature to the application .xls or .xla files, this step is not necessary.

2 **Unzip your application-Crystal Ball bundle by double-clicking or other procedure depending on how it is constructed.**

3 **Activate the installation and licensing scripts.**

4 **Launch the application, which should then launch Microsoft Excel and Crystal Ball.**

From then on, the application should be self-documenting with clearly labeled controls, text boxes, messages, and prompts to guide users through the application workflow.

Note: These instructions assume that your application sets a reference to cbdevkit.xla. To do this, start Microsoft Excel and select Tools, then Macros, and then Visual Basic Editor. Then, select Tools, then References and select cbdevkit.xla (you might have to browse for it in the folder where Crystal Ball is installed). Once you save your application file you should not have to do these steps again. In Microsoft Excel 2007, display the Developer tab as described in step 1 of the previous directions. Then, select Visual Basic, then Tools, and then References and check cbdevkit.xla.

Crystal Ball Runtime Licensing Requirements

Crystal Ball and related products must be used in accordance with the applicable Oracle license agreement.

A Using the OptQuest Developer Kit

In This Appendix

About the OptQuest Developer Kit

Welcome to the OptQuest Developer Kit for Crystal Ball Decision Optimizer. You can use the OptQuest Developer Kit to automate and control the setup and running of OptQuest optimizations from within programs written for a variety of Microsoft COM-compatible platforms. This opens up a range of possibilities:

- Integrating OptQuest with other software tools
- Creating turnkey applications that shield users from program intricacies
- Building custom reports or automating post-optimization analysis
- Setting up specialized optimization environments

The OptQuest Developer Kit provides a link between OptQuest and your program. It consists of a number of object-oriented development classes that control many aspects of OptQuest. Each copy of Crystal Ball Decision Optimizer comes enabled to use the OptQuest Developer Kit, so that programs you develop today can be run by other users with appropriate licenses.

The OptQuest Developer Kit provides a framework of classes that you can use with OptQuest to perform optimizations. It is a COM (Component Object Model) object, which allows the use of older languages that support ActiveX objects, such as Visual Basic 6.0 (VB6), C++, VBA, and others.

Who Should Use This Kit

The OptQuest Developer Kit is appropriate for advanced users who want to automate repetitive spreadsheet optimization processes. This book assumes that readers are familiar with a COM-compatible development platform and Crystal Ball with OptQuest.

What This Kit Includes

In addition to this book, the OptQuest Developer Kit includes the following files, available in the specified folders under the Crystal Ball installation folder (by default, C:\Program Files \Oracle\Crystal Ball):

- CBCOMDevKit.tlb (in the Bin folder), accessible directly and through CBDevKit.xla (in the Crystal Ball installation folder)
- OptDevKitRuntimeSample.xls (in the Examples folder), a sample spreadsheet with commented code so you can follow how the OptQuest Developer Kit was used to create several sample optimizations (see "OptDevKitRuntimeSample.xls" on page 264 for details)
- CB Com Devkit.chm (in the Docs folder), an online help file that describes the methods and properties within each class of the OptQuest Developer Kit (see "CB Com Devkit.chm" on page 262 for more information)

What You Will Need

Crystal Ball Decision Optimizer runs on several versions of Microsoft Windows and Microsoft Excel. For a complete list of required hardware and software, see the current *Crystal Ball Installation Guide*.

OptQuest Developer Kit Use and Structure

This section gives specific requirements for installing and using the OptQuest Developer Kit. It also summarizes the structure of the Developer Kit and introduces CB Com Devkit.chm, a help file that serves as an online reference to the classes, methods, properties, and other elements of the OptQuest Developer Kit.

Specific Requirements for Use

If you have a license to run OptQuest, you can use the OptQuest Developer Kit and run applications built with it. The following sections outline specific requirements for its installation and use.

Installing the OptQuest Developer Kit

If you are licensed to use OptQuest as part of Crystal Ball, the OptQuest Developer Kit is automatically available when you install Crystal Ball and is immediately available for your use. For installation instructions, see the current *Crystal Ball Installation and Licensing Guide*.

Microsoft .NET Framework Version

The OptQuest Developer Kit has been tested to run properly with Microsoft .NET Framework Versions 2.0, 3.0, and 3.5. You can force the OptQuest Developer Kit to run against a specific

version of the .NET Framework using an application configuration file in the form of "<*appname*>.exe.config" placed in the folder of the application's executable. The following section forces the OptQuest Developer Kit code to run against Microsoft .NET Framework 2.0:

```
<configuration>
  <startup>
    <supportedRuntime version="v2.0.50727" />
  </startup>
</configuration>
```

You might also want to run against a specific version of Microsoft Excel. In that case, *appname* would be excel and you would be modifying a file called excel.exe.config.

OptQuest Developer Kit Namespace

The OptQuest Developer Kit contains the following main namespace in Visual Basic code:

```
Decisioneering.CB.CBCOMDevKit
```

This namespace contains the classes required to set up and run an optimization with the OptQuest Developer Kit.

Important OptQuest Classes

Two categories of important classes in the OptQuest Developer Kit:

- "Optimization Setup Classes" on page 259
- "Optimization Runtime Classes" on page 260

For lists and descriptions of methods and properties in these classes, open CB Com Devkit.chm in the Docs folder beneath the main Crystal Ball installation folder (by default, C:\Program Files \Oracle\Crystal Ball). You can use this file as a reference at any time. For more information on this help system and how to use it, see "CB Com Devkit.chm" on page 262.

Optimization Setup Classes

The following are the main OptQuest Developer Kit classes used for setting up an optimization:

- OptSetup (main class for setting up optimizations)
- OptSetup.Constraint
- OptSetup.Decision
- OptSetup.Objective
- OptSetup.OptEfficientFrontier
- OptSetup.OptOptions
- OptSetup.Requirement

OptSetup

This class is the main class for setting up optimizations. You can use its methods and properties to access optimization variables—that is, to add, modify and delete variables and options for running optimizations.

OptSetup.Constraint

This class represents an OptQuest constraint and accesses constraint properties.

OptSetup.Decision

This class serves as a wrapper for Crystal Ball decision variables and accesses their properties.

OptSetup.Objective

This class is responsible for handling all operations related to OptQuest objectives.

OptSetup.OptEfficientFrontier

This class accesses OptQuest settings for Efficient Frontier analysis.

OptSetup.OptOptions

This class accesses OptQuest option settings, the preferences that control how long the optimization runs, whether it stops automatically, whether it runs deterministically or stochastically, and more.

OptSetup.Requirement

This class is responsible for handling all operations related to OptQuest requirements.

Optimization Runtime Classes

The following are the main OptQuest Developer Kit classes used to run optimizations and obtain results:

- OptRuntime (main class for running an optimization and obtaining results)
- OptRuntime.OptCounters
- OptRuntime.OptSolution
- OptRuntime.OptVarStatistics
- OptRuntime.OptVarSolutionFilter
- OptRuntime.ResultsWindow
- OptRuntime.RuntimeEfficientFrontier
- OptRuntime.TestPointInfo

OptRuntime

OptRuntime is the main class for running optimizations and obtaining results.

OptRuntime.OptCounters

This class is the set of optimization counters. You can use its members to count simulation iterations, Efficient Frontier test points, feasible solutions, and more.

OptRuntime.OptSolution

This class represents a single solution in an optimization, where a solution is a set of values for decision variables and their associated result values. Result values can be obtained for objectives, requirements, and constraints.

OptRuntime.OptVarStatistics

This class represents optimization variable statistics. These summarize the statistics for a single optimization variable in the current optimization. They are based upon the selected statistics filter.

OptRuntime.OptVarSolutionFilter

This class represents the optimization variable statistics filter. The statistics filter limits the set of solutions that affect the optimization variable statistics.

OptRuntime.ResultsWindow

This class represents the OptQuest Results window. Its methods and properties control the display of optimization results.

OptRuntime.RuntimeEfficientFrontier

This class determines if an Efficient Frontier analysis was run and provides a variety of runtime information about test points and results. (Efficient Frontier allows users to create a variable list of constraint and requirement right-hand-side values using a list of test points.)

OptRuntime.TestPointInfo

The methods and properties of this class summarize optimization information for a single Efficient Frontier test point.

Developing Optimization Code

This section discusses the development environment, resources available to help you learn and use the Optquest Developer Kit, and an overview of how to start and continue coding an optimization with illustrations from sample code.

Development Environment

For development with the OptQuest Developer Kit, you can use any development environment that supports the use of Microsoft COM components. Such an environment lets you use VB6, VBA (supplied with Microsoft Word and Microsoft Excel), C++, ASP, and similar COM-compatible languages (you will still need the Microsoft .NET Framework). To access the OptQuest Developer Kit, you must reference the COM object named CBCOMDevKit, which is registered on your computer during the installation.

Development Resources

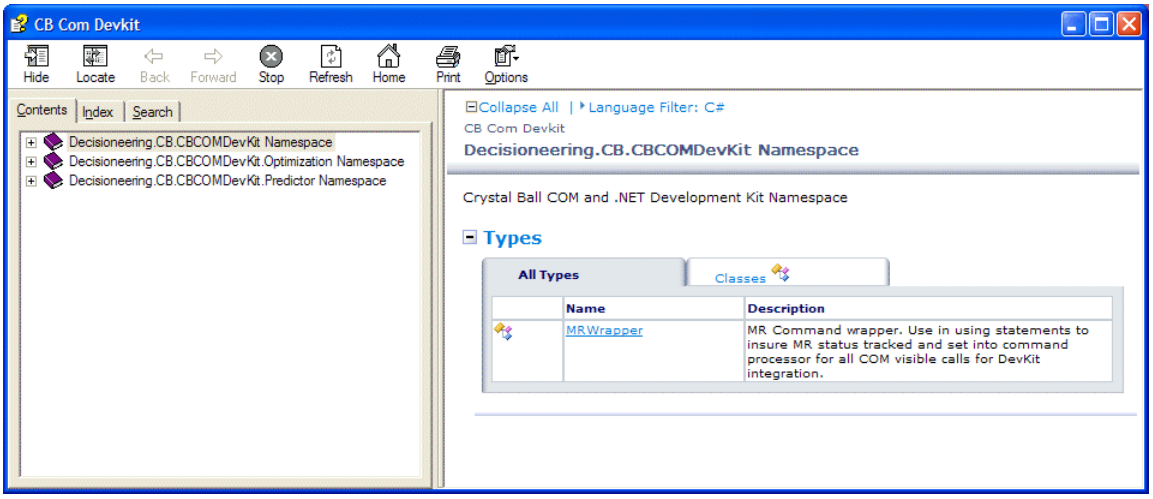
In addition to this book, the OptQuest Developer Kit contains two other files with information about kit contents and how to use it, CB Com Devkit.chm and OptDevKitRuntimeSample.xls.

CB Com Devkit.chm

This help file is a reference with lists and descriptions of methods and properties in OptQuest Developer Kit classes. It is located in the Docs folder beneath the main Crystal Ball installation folder (by default, C:\Program Files\Oracle\Crystal Ball\Docs). You can open and use CB Com Devkit.chm directly from that folder at any time.

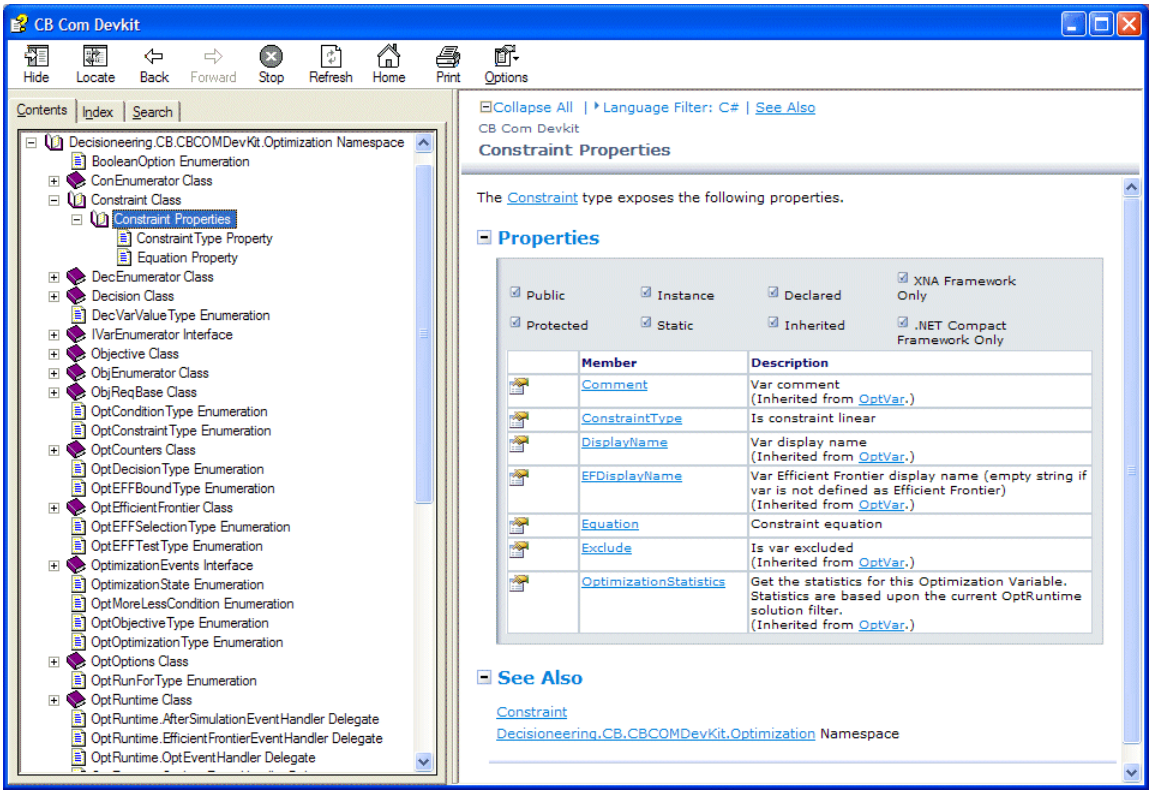
When you first open CB Com Devkit.chm, three namespaces appear in the navigation pane (Figure 1). These namespaces are only used in C# code, not Visual Basic (VBA) code. However, if you are using VBA, they indicate which classes are used for OptQuest automation (.Optimization) and which are used for Predictor (.Predictor). When writing VBA code, you omit .Optimization or .Predictor as shown in the code examples included in this Developer's Guide.

Figure 1 CB Com Devkit.chm with Three C# Namespaces



The Index and Search tabs are helpful for locating information about a specific class. For example, if you click the Search tab, type Constraint, and then click List Topics, you can select one of the listed topics in the navigation pane to display it in the content pane. See Figure 2, following, for an example of the Constraint Properties topic for OptQuest. You can also review lists and descriptions of methods, the classes themselves, and other important development information.

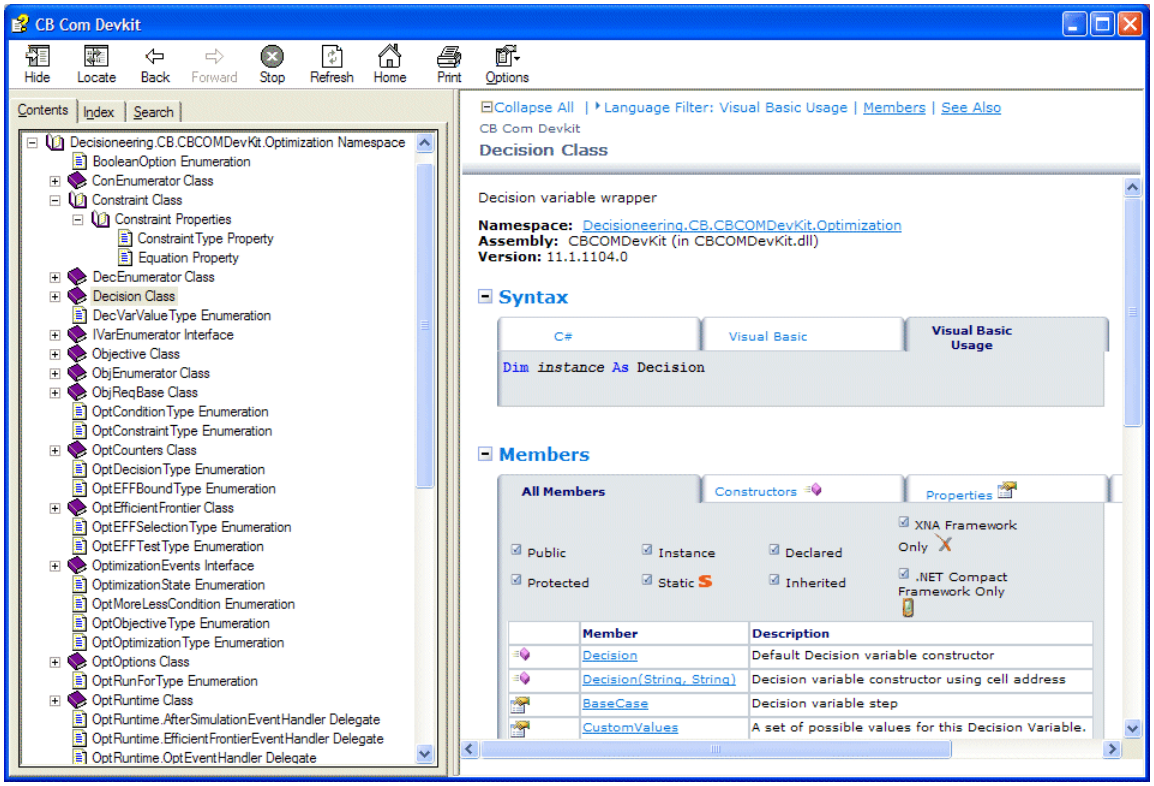
Figure 2 CB Com Devkit.chm Opened to the Constraint Properties Topic for OptQuest



CB Com Devkit.chm is generated from actual OptQuest Developer Kit code and has a development language filter to help you with your code. The default is C#. You can change to Visual Basic or Visual Basic Usage (for code snippet examples). The topic in Figure 2 has two collapsible headings, Properties and See Also. Other topics, such as Decision Class, shown in Figure 3, have an additional Syntax heading.

The Syntax tabs are linked to the Language Filter menu in the help window. In Figure 3, the language filter is set to Visual Basic Usage. Since the Visual Basic Usage filter shows sample Visual Basic code, a code snippet appears on the tab.

Figure 3 CB Com Devkit.chm opened to the Decision Class topic.



If you click the Visual Basic tab, Visual Basic appears in the Language Filter menu and the Visual Basic tab is selected in the Syntax tab group.

OptDevKitRuntimeSample.xls

This sample file contains code for two different optimization sequences, both based on the Portfolio Allocation example models. This file is located in the Examples folder beneath the main Crystal Ball installation folder. To use it, open OptDevKitRuntimeSample.xls in Crystal Balll. Choose to enable the macros. Then, open your development environment, for example, the Microsoft Excel Visual Basic Editor. In the project tree, select the class module PortfolioOptimizeClass. Commented sample code appears in the code window. You can run the code by clicking buttons on the Model worksheet. Then, you can examine the code and the comments to see how the samples were constructed. For more information about this sample, see "Coding an Optimization" on page 265 following.

Coding an Optimization

An optimization has three phases: building the model, setting up the optimization and then running it. The sample code in OptDevKitRuntimeSample.xls illustrates these steps with two sample optimizations. The first is based on Portfolio Analysis.xls, in the Crystal Ball Examples folder. The second adds Efficient Frontier analysis. If you haven't already, you might find it helpful to complete Tutorial 2 in the *Oracle Crystal Ball Decision Optimizer OptQuest User's Guide*. This tutorial explains how to complete the Portfolio Analysis.xls model, set up an optimization manually, and then run it. You will then understand the manual process and can better follow what the code is doing.

The following sections discuss optimization coding strategies:

Note: Before you begin coding, you should learn to perform optimizations with OptQuest. Many of the items used in OptQuest, such as constraints and objectives, match the corresponding objects in the OptQuest Developer Kit. See the *Oracle Crystal Ball Decision Optimizer OptQuest User's Guide* for complete descriptions of the optimization process and items.

Starting Out

The sample code in OptDevKitRuntimeSample.xls begins by explicitly declaring variable types for several variables:

```
Option Explicit
Private WithEvents PortfolioOptRuntime As CBCOMDevKit.OptRuntime
Private PortfolioOptSetup As CBCOMDevKit.OptSetup
Private ObjectiveTotal As CBCOMDevKit.Objective
Private RequirementRisk As CBCOMDevKit.Requirement
Private ConstraintFunds As CBCOMDevKit.Constraint
Private DecisionFund1 As CBCOMDevKit.Decision
Private PortfolioWorkbook As Workbook
Private InvestmentModel As Worksheet
Private Options As CBCOMDevKit.OptOptions
```

Next, it opens the subroutine that actually performs the optimization and defines error handling:

```
Sub RunOptimization(RunEF As Boolean)
    On Error GoTo ErrorHandling
```

From there, it creates (instantiates) Setup and Runtime objects and resets the optimization:

```
Set PortfolioOptSetup = New CBCOMDevKit.OptSetup
Set PortfolioOptRuntime = New CBCOMDevKit.OptRuntime
PortfolioOptRuntime.Reset
```

This subroutine goes on to:

- Close any open worksheets
- Create a new Portfolio Allocation workbook
- Clear any previous OptQuest data
- Select a primary workbook for optimization
- Set up the investments
- Set up the optimization
- Set up Efficient Frontier analysis, if requested
- Configure any options
- Run the optimization, show the Control Panel
- Get and display the optimization results
- Modify the OptQuest Results window display

Most of those operations are performed by calling subroutines. All code is heavily commented for easier understanding.

Continuing

For best results, examine all of the sample code. If something isn't clear, open CB Com Devkit.chm and search for the object in question. From there, experiment with your own code. You can begin by copying OptDevKitRuntimeSample.xls and modifying its code in different ways.

Error Handling

The OptQuest Developer Kit handles the same error codes as CB.MacroResult in the Crystal Ball Developer Kit. However, the OptQuest Developer Kit can take action when an error occurs (when a non-0 error code is generated). Error handling statements can take the form of `On Error GoTo` or `On Error Resume Next`.

For example, consider the following code sample from "OptDevKitRuntimeSample.xls" on page 264:

```
On Error GoTo ErrorHandling
```

This means that when an error condition occurs, the ErrorHandling code is called. Then, ShowError is called, and a series of messages describe the detected error. For details, look at the code in OptDevKitRuntimeSample.xls and search for ShowError. Also, look up CB.MacroResult and CB.MacroResultDetail in the *Crystal Ball Developer Kit User Manual*.

Accessing OptQuest with the Crystal Ball Developer Kit

The object instantiation in "OptDevKitRuntimeSample.xls" on page 264 uses the Set ... = New syntax to create objects in the CBCOMDevKit namespace:

```
Set PortfolioOptSetup = New CBCOMDevKit.OptSetup
Set PortfolioOptRuntime = New CBCOMDevKit.OptRuntime
```

You can also create and use these objects outside of that namespace with the following Crystal Ball Developer Kit calls:

- CB.OptSetup
- CB.OptRuntime

To do this, you must reference CBDevKit as well as CBCOMDevKit. Then, you can use Visual Basic or VBA syntax to apply methods and properties. For example, CB.OptSetup.Show can be used in Visual Basic or VBA code to display the OptQuest wizard.

User-defined Event Macros for Optimizations

This section summarizes information about user-defined Microsoft Excel VBA event macros that can run at various times while an OptQuest optimization runs.

User-defined Event Macro Names

Users can create Microsoft Excel VBA event macros for workbooks. Depending on the event selected, Crystal Ball runs them automatically at various points in a simulation or optimization. The original simulation-specific macros are described in Chapter 4 of the *Crystal Ball User Manual*.

Beginning with Crystal Ball 11.1.1.0.00, macros based on VBA events are now available to run at various points during an OptQuest optimization. Table 200, following, summarizes events that can be defined with VBA code and describes when they run.

Note: Event macros must be attached to worksheets or workbooks and not modules.

Table 200 Events Accessible through VBA and When They Run within an Optimization

Event	Type	Runs...
BeforeOptimizationEvent	Optimization	At the start of an optimization, when the OptQuest Run command is given or an equivalent command is given using the OptQuest Developer Kit.
BeforeIterationEvent	Optimization	At the start of each iteration of an optimization, just after the decision variable values for that iteration have been written to the workbook, but before any nonlinear constraints have been evaluated by OptQuest.
BeforeSimulationEvent	Simulation	On each iteration of an optimization just before the simulation begins. At this point, nonlinear constraints have been evaluated by OptQuest.
AfterSimulationEvent	Simulation	On each iteration of an optimization, just after the simulation completes. Any extracted data values will have been written to the workbook as part of the simulation processing. Forecast objective and requirement values have not yet been updated to OptQuest. **Note:** at this point any solution values for this iteration are not yet valid.

Event	Type	Runs...
AfterIterationEvent	Optimization	After each iteration of an optimization. The simulation is complete, any extract data has been written to the workbook and objective and any requirement values have been updated to OptQuest. **Note:** at this point all solution values for this iteration are valid.
AfterOptimizationEvent	Optimization	After an optimization is complete.
EfficientFrontierEvent	Optimization	After each test point in an Efficient Frontier optimization is complete.

The BeforeOptimizationEvent macro runs at the beginning of the optimization, when the Run button is first clicked or the Run command is given. After that, the Simulation and Iteration event macros take over and run as described until all requested trials and simulations have been run (or a stopping point set in the Advanced Options dialog has been reached). At that point, the optimization is complete and the AfterOptimizationEvent macro runs.

Using Events in VBA

To use these events in Microsoft Excel VBA, open the Microsoft Excel Visual Basic Editor and open a code window for a workbook.

Begin by explicitly declaring the CBCOMDevKit.OptRuntime class as a variable with events:

```
Private WithEvents PortfolioOptRuntime As CBCOMDevKit.OptRuntime
```

In this example, the variable name PortfolioOptRuntime is declared as on OptRuntime class type.

Then, declare other variables as required. For more information see "Coding an Optimization" on page 265.

Whenever you want to enter an event macro into your code, go to the first dropdown list in the code window and choose the name of the variable assigned to OptRuntime.

Available events are listed in the second dropdown list in the code window. When you click one, it is inserted into your code for that workbook as a private subroutine, for example:

```
Private Sub PortfolioOptRuntime_AfterIterationEvent(IterationsCompleted As Long)
End Sub
```

The subroutine takes the name of the variable assigned to OptRuntime followed by an underscore and the name of the selected event. Any required and optional parameters appear in parentheses following the name.

You insert code as usual, using any appropriate VBA code and macros from the Crystal Ball and OptQuest Developer Kits. For example, you can set cell values or perform other operations described elsewhere in this Overview document or the *Crystal Ball Developer Kit User Manual*.

Then, when the selected event occurs, the code runs.

Important Note: In any event handler that modifies values or formulas within a Microsoft Excel workbook (or calls functions that could modify values), you should insert the following code:

```
'Indicate this event was handled
```

```
PortfolioOptRuntime.EventHandled
```

where PortfolioOptRuntime is an OptRuntime object. This code informs the optimization engine that events are being handled and allows correct processing in the case where event handlers modify values in the Microsoft Excel workbook. It does not matter where the code is placed as long as it is within the event handler.

Event Signatures

The following are signatures for each of the event macros in OptRuntime:

- "BeforeOptimizationEvent" on page 269
- "BeforeIterationEvent" on page 269
- "BeforeSimulationEvent" on page 269
- "AfterSimulationEvent" on page 269
- "AfterIterationEvent" on page 270
- "AfterOptimizationEvent" on page 270
- "EfficientFrontierEvent" on page 270

The subroutine names shown here before the underscore will change depending on the variable name assigned to OptRuntime.

BeforeOptimizationEvent

```
Private Sub PortfolioOptRuntime_BeforeOptimizationEvent()
   'Your code is here
End Sub
```

BeforeIterationEvent

```
Private Sub PortfolioOptRuntime_BeforeIterationEvent()
   'Your code is here
End Sub
```

BeforeSimulationEvent

```
Private Sub PortfolioOptRuntime_BeforeSimulationEvent()
   'Your code is here
End Sub
```

AfterSimulationEvent

```
Private Sub PortfolioOptRuntime_AfterSimulationEvent( _
    IterationsCompleted As Long, _
    Trials As Long, _
```

```
    MaxTrials As Long, _
    Progress As Double, _
    WasAborted As Boolean, _
    WasStoppedOnError As Boolean, _
    WasStoppedOnPrecision As Boolean, _
    HasNewBestSolution As Boolean)
   'Your code is here
End Sub
```

In this signature, parameters or arguments are defined as follows:

- IterationsCompleted = Number of iterations completed
- Trials = Number of trials completed for this simulation
- MaxTrials = Number of trials scheduled to run (total trials per simulation)
- Progress = Percent of trials completed (expressed as a decimal), 0 to 1.0
- WasAborted = True if manually stopped
- WasStoppedOnError = True if stopped on error
- WasStoppedOnPrecision = True if stopped on precision control
- HasNewBestSolution = New value of objective; for optimizations, was this a new best solution

AfterIterationEvent

```
Private Sub PortfolioOptRuntime_AfterIterationEvent(IterationsCompleted As Long)
   'Your code is here
End Sub
```

In this signature, parameters or arguments are defined as follows:

- IterationsCompleted = Number of iterations completed

AfterOptimizationEvent

```
Private Sub PortfolioOptRuntime_AfterOptimizationEvent(IterationsCompleted As Long)
   'Your code is here
End Sub
```

In this signature, parameters or arguments are defined as follows:

- IterationsCompleted = Number of iterations completed

EfficientFrontierEvent

```
Private Sub PortfolioOptRuntime_EfficientFrontierEvent(TestValue As Double)
   'Your code is here
End Sub
```

In this signature, parameters or arguments are defined as follows:

- TestValue = The Efficient Frontier test value just completed

Note: Note: Returning anything other than a zero from a macro will stop the optimization.

Constraints and Macros

You can set up an OptQuest constraint in the form `A1 <= 100` or simular and use CBBeforeSimulation or CBBeforeIteration to modify cell A1. If you do this, a "constant constraint" warning appears and warns you that one or more constraints do not appear to contain decision variables, that they are marked as constants, and therefore no feasible solutions may be found.

This warning only applies if you don't have a macro that modifies the cell. If you are using a macro to change values in the referenced cell, you can close the warning and continue with the optimization. Then, the constraint will work as intended. If you check the "Do not show" box, you won't need to respond to the message again while you are using that particular model.

Global Macros

If you want to run certain event macros with any or all of your Crystal Ball Decision Optimizer models, you need to make sure that the workbook containing those event macros is open in Microsoft Excel.

B Using the Predictor Developer Kit

In This Appendix

About the Predictor Developer Kit

Note: The Predictor Developer Kit described in this Developer's Guide is completely rewritten. Developer kit code written for automating CB Predictor in Crystal Ball versions earlier than 11.1.1.3.00 is not compatible with the current version of Predictor or this Predictor Developer Kit.

Welcome to the Predictor Developer Kit for all Crystal Ball products. You can use the Predictor Developer Kit to automate and control the setup and running of Predictor time-series forecasts from within applications written for a variety of Microsoft COM-compatible platforms. This opens up a range of possibilities:

- Integrating Predictor with other software tools
- Creating turnkey applications that shield users from program intricacies
- Building custom reports or automating analysis when time-series forecasts are complete
- Setting up specialized time-series-forecast environments

The Predictor Developer Kit provides a link between Predictor and your application. It consists of a number of object-oriented development classes that control many aspects of Predictor. Each Crystal Ball product comes enabled to use the Predictor Developer Kit, so that applications you develop can be run by other users with appropriate licenses.

The Predictor Developer Kit provides a framework of classes that you can use with Predictor to perform time-series forecasts. It is a COM (Component Object Model) object, which allows the use of older languages that support ActiveX objects, such as Visual Basic 6.0 (VB6), C++, VBA, and others.

Who Should Use This Kit

The Predictor Developer Kit is appropriate for advanced users who want to automate repetitive spreadsheet time-series forecasts. This documentation assumes that readers are familiar with a COM-compatible development platform and Crystal Ball with Predictor.

What This Kit Includes

In addition to this Developer's Guide, the Predictor Developer Kit includes the following files, available in the specified folders under the Crystal Ball installation folder (by default, C:\Program Files\Oracle\Crystal Ball):

- CBCOMDevKit.tlb (in the Bin folder), accessible directly and through CBDevKit.xla (in the Crystal Ball installation folder)
- Toledo Gas.xls (in the Examples folder), a sample spreadsheet with commented code so you can follow how the Predictor Developer Kit was used to create several sample time-series forecasts
- CB Com Devkit.chm (in the Docs folder), an online help file that describes the methods and properties within each class of the Predictor Developer Kit (see "CB Com Devkit.chm" on page 262 for more information)

Developer Kit Use and Structure

This section gives specific requirements for installing and using the Predictor Developer Kit. It also summarizes the structure of the Developer Kit and introduces CB Com Devkit.chm, a help file that serves as an online reference to the classes, methods, properties, and other elements of the Predictor Developer Kit.

Specific Requirements for Use

If you have a license to run Oracle Crystal Ball, Fusion Edition, Crystal Ball Decision Optimizer, or Oracle Crystal Ball Enterprise Performance Management, Fusion Edition, you can use the Predictor Developer Kit and run applications built with it. The following sections outline specific requirements for its installation and use.

Installing the Predictor Developer Kit

The Predictor Developer Kit is automatically available when you install any of the Crystal Ball products, version 11.1.1.3.00 or later, and is immediately available for your use. For installation instructions, see the current *Oracle Crystal Ball Installation and Licensing Guide*.

Microsoft .NET Framework Version

The Predictor Developer Kit has been tested to run properly with Microsoft .NET Framework Versions 2.0, 3.0, and 3.5. You can force the Predictor Developer Kit to run against a specific version of the .NET Framework using an application configuration file in the form of "<*appname*>.exe.config" placed in the folder of the application's executable. The following section forces the Predictor Developer Kit code to run against Microsoft .NET Framework 2.0:

```
<configuration>
  <startup>
    <supportedRuntime version="v2.0.50727" />
  </startup>
</configuration>
```

You might also want to run against a specific version of Microsoft Excel. In that case, *appname* would be excel and you would be modifying a file called excel.exe.config.

Predictor Developer Kit Namespace

The Predictor Developer Kit contains the following main namespace:

`Decisioneering.CB.CBCOMDevKit`

This namespace contains the classes required to set up and run time-series forecasts with the Predictor Developer Kit.

Note: "CB Com Devkit.chm" on page 262 describes a reference help file that summarizes all the OptQuest and Predictor Developer Kit objects and shows the hierarchy of the development system. When you first open CB Com Devkit.chm, three namespaces appear in the navigation pane (Figure 1). These namespaces are only used in C# code, not Visual Basic (VBA) code. However, if you are using VBA, they indicate which classes are used for OptQuest automation (.Optimization) and which are used for Predictor (.Predictor). When writing VBA code, you omit .Optimization or .Predictor as shown in the code examples included in this Developer's Guide.

Important Predictor Classes

Two categories of important classes in the Predictor Developer Kit:

- "Predictor Setup Class" on page 276
- "Predictor Results Class" on page 277

Other classes support those two areas of functionalilty.

For lists and descriptions of methods and properties in Predictor namespace classes, open CB Com Devkit.chm in the Docs folder beneath the main Crystal Ball installation folder (by default, C:\Program Files\Oracle\Crystal Ball). You can use this file as a reference at any time. For more information on this help system and how to use it, see "CB Com Devkit.chm" on page 262.

If you select a class in the navigation pane, its structure is outlined in the other pane, with annotations. Click any underlined link to learn more about that object.

Classes support functionality in the Predictor user interface (UI). To help you locate appropriate classes, class names are designed to reflect the UI.

Predictor Setup Class

Main classes for setting up, validating, and running Predictor:

- PredSetup (main class interface for setting up and running Predictor)
- PredSetup.PredInputData
- PredSetup.PredDataAttributes
- PredSetup.PredMethods
- PredSetup.PredOptions
- PredSetup.PredSeries

PredSetup

This is the top level interface class that provides access to objects to set and define all Predictor settings available through the Predictor wizard user interface. The properties of this class enable access to classes that correspond to the wizard panels: Input Data, Data Attributes, Methods, and Options. The methods of this class enable validation of input data and settings and also provide a method to run the time series forecast.

PredSetup.PredInputData

This class provides access to input data properties corresponding to the Predictor wizard Input Data panel. This class enables you to control items such as input data range, data orientation, and headers.

PredSetup.PredDataAttributes

This class provides access to data attribute settings corresponding to the Predictor wizard Data Attributes panel. This class enables you to control items such as seasonality, data screening, and outlier detection.

PredSetup.PredMethods

This class provides access to methods settings corresponding to the Predictor Methods panel. This class enables you to control which time series and regression methods to use and to set method parameters.

PredSetup.PredOptions

This class provides access to Predictor options corresponding to the Predictor Options panel. Use this class to specify options such as error measure and forecasting technique.

PredSetup.PredSeries

This class provides access to setup information for each series. Use this class to manage individual input series information such as series name, Microsoft Excel number format, seasonality and cycle, regression variable details, statistics, and access series data values. An enumerator property of the PredSetup class (.Series) enables you to access each series by index into the input data range.

Predictor Results Class

Main classes for accessing Predictor results:

- PredResults (main class interface for accessing Predictor results)
- PredResults.PredResultsPreferences
- PredResults.PredResultsWindow
- PredResults.PredSeriesResult

PredResults

This is the top level interface class that provides access to objects that retrieve results available through the Predictor results form. This class has properties to access results preferences and to select series and methods. This class has a method to paste the forecast results to your model.

PredResults.PredResultsPreferences

This class provides access to results preferences such as confidence intervals, number of periods to forecast, and Predictor results window display options.

PredResults.PredResultsWindow

This class provides access to the Predictor results window and enables you to set items such as selected series and method. This class enables you to control the Predictor results dialog and access all of the controls on the form.

PredResults.PredSeriesResult

This class provides access to per series result information. This class provides access to information such as series name, best fit method, forecast adjustments, result statistics, series historical data values and forecast data values. An enumerator property of the PredResults class (.Series) enables you to access each series by index into the input data range.

Developing Time-Series Forecasting Code

For development with the Predictor Developer Kit, you can use any development environment that supports the use of Microsoft COM components. Such an environment lets you use VB6, VBA (supplied with Microsoft Word and Microsoft Excel), C++, ASP, and similar COM-compatible languages (you will still need the Microsoft .NET Framework). To access the Predictor Developer Kit, you must reference the COM object named CBCOMDevKit, which is registered on your computer during the installation.

In addition to this book, the Predictor Developer Kit contains two other files with information about kit contents and how to use it, CB Com Devkit.chm and Toledo Gas.xls. CB Com Devkit.chm is a reference help file with lists and descriptions of methods and properties in Predictor Developer Kit classes. It also contains information on the OptQuest Developer Kit, available with Oracle Crystal Ball Decision Optimizer, Fusion Edition.

For more information about CB Com Devkit.chm, see "CB Com Devkit.chm" on page 262. Although the information there refers to the OptQuest Developer Kit, the information is basically the same for the Predictor Developer Kit. Be aware that class names are different, as described in "Important Predictor Classes" on page 275.

Toledo Gas.xls is a Crystal Ball example file used as a tutorial example in the *Oracle Crystal Ball Predictor User's Guide*. It contains annotated code that performs similar tasks to those described in the Predictor User's Guide.

➤ To view the Predictor Developer Kit sample code:

1. **Open Toledo Gas.xls.**

   From the Window task bar, you can select **Start**, then **All Programs**, then **Oracle Crystal Ball**, then **Examples**, and then **View Examples Guide**. From the list of examples, select **Toledo Gas**.

2. **With the example model open, open the Microsoft Excel Visual Basic Editor.**
3. **In the Editor, select Tools, and then References. Confirm that CBCOMDevKit is selected in the list.**
4. **In the Project Explorer, select Module1 for Toledo Gas.xls.**
5. **Study the code annotations for examples of how to perform basic Predictor operations.**

   Notice that code illustrating the event feature and ARIMA time-series forecasting is included.

As you study the code annotations, refer to basic code strategies outlined in "Coding an Optimization" on page 265. Although this information refers to the OptQuest Developer Kit, basic rules and guidelines apply to the Predictor Developer Kit.

Note: Although coding is similar in the OptQuest and Predictor Developer Kits, ignore the sections on user-defined event macros and events in VBA. The Predictor Developer Kit does not support user-defined events.

Index

D

E

F

R

S

T

U

V

W