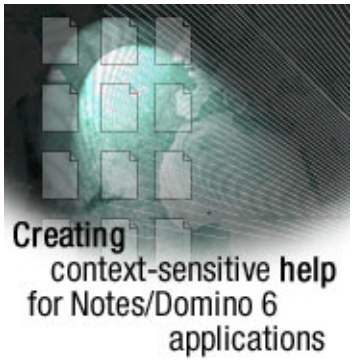
Lotus Developer Domain
The Technical Resource for Lotus software
LDD Today




**Level:** All
**Works with:** Domino Designer 6
**Updated:** 01-Oct-2002

by
Susanna
Doyle

For Notes R5, Joann Spera's article "**Creating Context-Sensitive Help for Applications**" described how to use the HelpRequest event to make the elements (forms, views, folders, and pages) in your Domino application present the topic of your choice when users of your application choose Help - Context Help or press F1.

The good news is that all the functionality described in that article works in Notes/Domino 6. And the better news is, context-sensitive help for your applications has now been enhanced in several ways:

- Context-sensitive help now works for dialog boxes in your applications that are generated from forms, subforms, or views using either @DialogBox, the LotusScript DialogBox method, @Picklist, or the LotusScript PickListStrings method.
- The Help event has been extended to subforms. It's now called onHelp on forms, subforms, and page, and still called HelpRequest on views and folders. Its existence on subforms lets you present help for dialog boxes created with the subforms.
- The onHelp event in Domino Designer 6 lets you choose between JavaScript, LotusScript, and the formula language to present help. For example, in Internet Explorer, using JavaScript, you can redirect F1 so that it opens your help instead of Explorer's help.
- The new @function @GetFocusTable lets you present help specific to the current row or column of a table, or the current tab of a tabbed table. This is useful if your application uses tabbed tables in or out of dialog boxes.
- The new @function @GetField lets you present help based on a field value in a document that has not yet been saved. This is useful to users who are in the process of filling out a form and need help before they have saved any data.

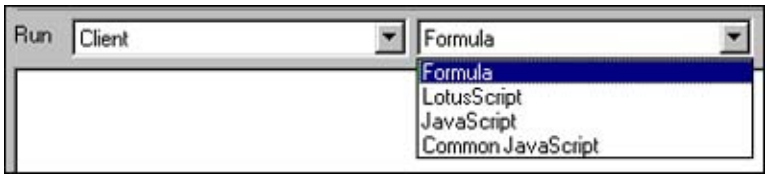
This article assumes that you are familiar with the basics (described in the first **article**) of adding @Command[OpenHelpDocument] formulas to events in your application. This @command has not changed for Domino Designer 6, and still remains the simplest way to open a Help topic. As before, you can create your Help topics directly in the database for your application, adding a Help form and view, or you can create a separate Help database using the Help example database we provide.

The Help example database's design and how-to documents provided with the first article have all been updated to match the improved design of the Notes 6 Help. We've also enhanced the sample Discussion database provided with the first article, and added a Discussion (Integrated) database, in which help is part of the database in a hidden view instead of in a separate database. You can download all three of these updated databases from the **Sandbox**. Be sure to save the databases in your \data folder of whatever directory contains your installation of Notes 6.

## Understanding the new onHelp event

When you design a form, subform, or page in Domino Designer 6 and select the onHelp event, you see two drop-down menus that let you specify code for the event to run in the Notes client or in Web browsers.




For the Notes client, you can use either the formula language, LotusScript, or JavaScript. If you select Common JavaScript, the JavaScript you enter will be used for both the Notes client and the Web browser.

For the Web, you can select either JavaScript or Common JavaScript. As mentioned, Common JavaScript will be used for both Notes and the Web. You can use JavaScript to open a topic in your Help database, but only if you make the database available over the Web from a Domino server and only in Internet Explorer. For more information, see **Using JavaScript on the Help event in Internet Explorer** later in this article.

The HelpRequest event on views and folders is the same as in Notes R5; you can use only the formula language for it.

## Help for dialog boxes in your application

If you are familiar with @DialogBox and @Picklist, you know that you can present custom dialog boxes in your application that show the contents of a layout region or table in a form or subform, or display a view complete with its column headings and scroll bar.

In Domino Designer 6, if you add code to the onHelp event for the form or subform that supports such a dialog box, or to the HelpRequest event of a view that supports one, your code will be executed when the dialog box is opened and users press a Help key combination. In forms supported by dialog boxes, users can also click the question mark icon to request help.

For example, click the Register button in the sample Discussion database's All Documents view to see the following dialog box:


Next, press F1 or click the question mark icon to see the topic shown below. The Help code executes whether your Help topics are stored in a separate, dedicated Help database or stored in a view in your application. When a dialog box is open, as you may know from using the product Help in Notes, the Help topic appears on its own in a small rectangular window. This window does not contain the frameset with the Contents, Index, and Search views that you normally see in the separate Help database. It has three links—Go Back, Print, and Done—so that users can follow hyperlinks throughout your Help topics if you have them, print the topic, or close the small Help window.

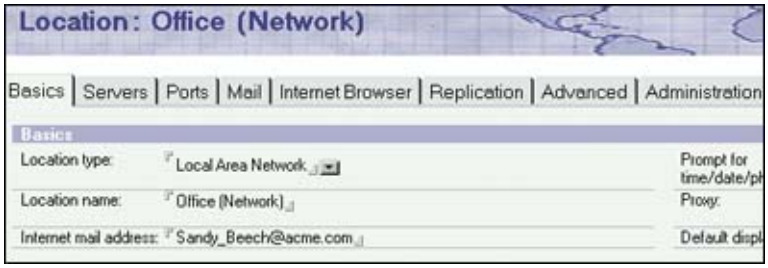

 

Note that if you have a subform that supports a dialog box and that subform also appears on a form that supports a different dialog box, the onHelp event on the form always "wins" over the onHelp event on the subform when users press F1. For best results when structuring your Help, use the subforms only directly in dialog boxes, not as part of other forms, if you want to use them to support dialog boxes. Also, the onHelp event for a view can be accessed when you use the LotusScript PickListStrings method with the PICKLIST_CUSTOM parameter to select documents from a view. It is not supported with the other parameters for the method.

## Help for tabbed tables

Ever since Domino Designer added the feature of displaying a table with tabs, designers have used tabbed tables both in and out of dialog boxes to present varied information to users.

For example, in Lotus Notes 6, choose File - Mobile - Edit Current Location. The form you see is laid out with up to nine tabs (depending on the type of Location you're currently using). The following illustration shows a Local Area Network type of connection with eight tabs.




Many of the tabs on this form are set to display context-sensitive help, using a formula on the onHelp event. If you press F1 while on the Advanced tab, you see the Help topic "Advanced settings for a location." If you then click the Mail tab on the top level of the form and press F1, you'll see a different topic: "Mail settings for a location."

The formula that makes this possible uses a new @function called @GetFocusTable. This function returns three items: the name of the current table, and the most recently clicked row and column in the table.

You could use this information to provide Help for any table, of course, but it's most useful for tabbed tables, where the row represents the tab and the content on other tabs (rows) is out of sight when the user requests help. @Command([OpenHelpDocument]) can open a different Help topic for each condition, but it can't, for example, jump to a midtopic point in a large topic, which might be more appropriate documentation for an ordinary table.

To see a tabbed table onHelp formula working, open the sample Discussion database and click the Register action button. In the dialog box, click different tabs and press F1 for each.

The dialog box is supported by the hidden form (RegisterDlg). The onHelp formula for that form is as follows:

```
tablename := @GetFocusTable([TableName]);
row := @GetFocusTable([CellRow]);
@If(
tablename = "RegisterMain";@Command([OpenHelpDocument]; "discdoc6.nsf";"(Help)";
"F_REGISTER");
(tablename = "Preferred" & row = "1");@Command([OpenHelpDocument]; "discdoc6.nsf";"(Help)";
"F_REGISTER_shrubs");
(tablename = "Preferred" & row = "2");@Command([OpenHelpDocument]; "discdoc6.nsf";"(Help)";
"F_REGISTER_annuals");
(tablename = "Preferred" & row = "3");@Command([OpenHelpDocument]; "discdoc6.nsf";"(Help)";
"F_REGISTER_perennials");
(tablename = "Preferred" & row = "4");@Command([OpenHelpDocument]; "discdoc6.nsf";"(Help)";
"F_REGISTER_ground");
@Command([OpenHelpDocument]; "discdoc6.nsf";"(Help)"; "F_REGISTER"))
```

A useful way to debug formulas like this is to add an @prompt line to show which table and row has the focus. Try adding this @Prompt line (shown in bold) to the formula on the onHelp event:

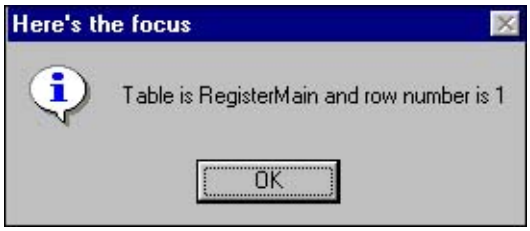
```
tablename := @GetFocusTable([TableName]);
```

```
row := @GetFocusTable([CellRow]);
@Prompt([ok];"Here's the focus";"Table is "+tablename+" and row number is "+row);
@If(
tablename = "RegisterMain";@Command([OpenHelpDocument]; "discdoc6.nsf";"(Help)";
"F_REGISTER");
(tablename = "Preferred" & row = "1");@Command([OpenHelpDocument]; "discdoc6.nsf";"(Help)";
"F_REGISTER_shrubs");
(tablename = "Preferred" & row = "2");@Command([OpenHelpDocument]; "discdoc6.nsf";"(Help)";
"F_REGISTER_annuals");
(tablename = "Preferred" & row = "3");@Command([OpenHelpDocument]; "discdoc6.nsf";"(Help)";
"F_REGISTER_perennials");
(tablename = "Preferred" & row = "4");@Command([OpenHelpDocument]; "discdoc6.nsf";"(Help)";
"F_REGISTER_ground");
@Command([OpenHelpDocument]; "discdoc6.nsf";"(Help)"; "F_REGISTER"))
```

Save and close the form, return to the view in the Notes client, click the Register button, and press F1.

You see a message box like this:




When you close the box, the Help opens.

This @Prompt can be useful to figure out what your @If formula is doing. For example, if you add many nested tables, you may find some of your conditional statements must become a bit more complex, like this one in the formula for the Server form on the Domino Directory:

```
(@Contains(tablename;"Ports_Proxies") | (tablename = "Server_Ports_Tabs" & row = "3"))
```

This condition opens a particular Help topic if either the name of the table that has focus contains the string Ports_Proxies or the name of the table is Server_Ports_Tabs and the third row of that table has the focus. The tables on the Server form are named using variations on similar strings. This allows the formula to provide a general topic that covers several tabs, or if the condition of a certain tablename and row number is met, to provide a more specific topic.

If you are a Domino administrator, several important forms in the Domino Directory have context-sensitive Help for many different tabs in many tables. For example, take a look at the Server, Connection, and some of the Policy forms.

When you finish debugging your formula, remove the @prompt line.

Note that if your formula fails because a string is missing or incorrect in the AppSpecID field in a Help topic that is supposed to open, you see a different message, like this:


## Help based on a field value in an unsaved document

When Help events were added in Notes R5, you could provide help for a form, but the formula you used could test

field values only if the document the user had open had been saved. Now, in Notes/Domino 6, a new function, @GetField, lets you work with default values of fields.

The Domain form in the Domino Directory contains a good example of @GetField in its onHelp formula:

```
domtype:=@GetField("DomainType");

HelpDoc := @If(
        domtype = "ForeignDomain";                  "pubnames_f_Domain_foreign";
        domtype = "NonAdjacentDomain";  "pubnames_f_Domain_nonadj";
        domtype = "AdjacentDomain";                 "pubnames_f_Domain_adj";
        domtype = "DomainRule";             "pubnames_f_Domain_for_x400";
        domtype = "SMTPDomain";                     "pubnames_f_Domain_for_smtp";
        domtype = "ccMailDomain";                   "pubnames_f_Domain_for_ccmail";
        domtype = "GlobalDomain";                   "pubnames_f_Domain_glob";
        "pubnames_f_Domain");

@Command([OpenHelpDocument]; [AdminHelp]; "(Help)"; HelpDoc)
```

Though the form does have several tabs, it makes more sense to determine the Help that appears for it based on the type of domain the administrator is configuring. The formula therefore uses @GetField to determine which value is current in the drop-down list provided by the DomainType field. If an administrator composes a new Domain form and changes the type of domain, pressing F1 for the different types provides different Help.

## Using JavaScript on the Help event in Internet Explorer

You can use the following JavaScript on the onHelp event of a form or page to open your Help database in Microsoft Internet Explorer (5.5 or later). This allows you to present all your Help through a Web browser, whether your application works in the Notes client, in the Web browser, or both.

```
noteshelpwindow = window.open('http://localhost/discdoc6.nsf/(Help)/Main_F',
'helpwin','toolbar=0,location=0,directories=0,status=0,menubar=0,scrollbars=1,resizable=0,
titlebar=0,left=200,width=700,height=400');
return false
```

This script creates a window in the browser and opens the URL into the window. The return false statement stops the browser from going on to open the Explorer help. The localhost element opens the URL from the database in your local directory. If you plan to place your Help database on a Domino Web server, replace "localhost" with the URL for your server and directory, for example

```
//YOURSERVER.YOURCOMPANY.com/help
```

You cannot use this JavaScript on the onHelp events of the topic or response forms in the sample Discussion database because some JavaScript already in those forms conflicts with this script. However, you can try this JavaScript on a page element we've provided in the sample Discussion design. In the All Documents view, click the Test Browser Help button.

## Adding integrated Help topics to your application

The previous article, **Creating Context-Sensitive Help for Applications**, contained steps for adding a Help form and view to your application but provided only a sample of a separate Help database.

A new sample we've provided—Discussion (Integrated)—has a Help form and view built into its design to handle Help requests. The Help view is not hidden, but you could hide it if you wanted all Help for your application to be accessed only through F1.

Note that the window that Notes uses for help requests when help is integrated in your application does not allow the full help presentation with the Contents, Index, and Search action bar. If you want these features, your Help must be in a separate database.

The @Command formula for opening Help topics from the same database can use @Subset to address the same server and the same database:

```
@Command([OpenHelpDocument];@Subset(@DbName;1):@Subset(@DbName;-1);"(Help)";
```

"Main_F")

## Using LotusScript on the onHelp event

You can open a Help document with LotusScript, but you can't use the special Help window. You must be in a form or page to use the following LotusScript on onHelp; most LotusScript does not work on F1 from an open dialog box.

In the Discussion (Integrated) example database, choose Create - Interest Profile A and press F1. A help topic is opened by the following LotusScript on the onHelp event:

```
Dim s As New NotesSession
Dim ws As New NotesUIWorkspace
Dim db As NotesDatabase
Set db = s.CurrentDatabase
Dim view As NotesView
Dim doc As NotesDocument
Set view = db.GetView("(Help)")
Set doc = view.GetFirstDocument
While Not (doc Is Nothing)
        Dim item As NotesItem
        Set item = doc.GetFirstItem("AppSpecID")
        If item.Text = "Main_F" Then
                Dim unid As String
                unid = doc.UniversalID
                Set doc = db.GetDocumentByUNID(unid$)
                Call ws.EditDocument(False, doc, True,,False)
        Else
        End If
        Set doc = view.GetNextDocument(doc)
Wend
```

For the purpose of simply opening a specific Help document, as you can see, @Command([OpenHelpDocument]) is easier and more powerful than LotusScript. There is no LotusScript equivalent for this @Command, and as an @Command it cannot be run as part of a script using the Evaluate statement.

However, you may want to use LotusScript on the Help event to do something quite different that LotusScript can do more efficiently than can the formula language. What you do depends on your imagination and level of familiarity with LotusScript.

Discussion (Integrated) includes three other examples of the Interest Profile form. Create each one, press F1 to trigger its onHelp event, and see what the LotusScript on the event does.

## Conclusion

We hope this article gives you some ideas for adding Help to your application. With new features in Domino 6 and Notes 6, you have even more power to customize your Help presentation. Once users realize that pressing F1 leads them to the correct Help topic for the dialog box they're in, or the tab on a table that they're reading, as well as for whatever form or view they use, they should feel much more comfortable with your application. And their productivity should also be much improved.