by
Joann
Spera

Level: Intermediate
Works with: Designer 5.0
Updated: 07/01/99

Inside this article:

Related links:

Sample help system download

Domino and Notes Release 5 Help Templates

R5 templates with context-sensitive help sidebar

Get the PDF:
Kb

Get Acrobat Reader

You've been asking for it, and now R5 gives it to you -- context-sensitive help for applications. For those unfamiliar with the term, *context-sensitive help* means that when a user presses F1 while performing a task in your application, Notes displays help that pertains to the current task.

R5 provides features that let you display the context-sensitive help topics you write for applications. R5 also provides context-sensitive help for many of the system templates. This means that you do not have to write help for the commonly-used templates, such as Mail and the Personal Address Book, so you can concentrate your efforts on templates that you design and develop. For a complete list of system templates that have context-sensitive help enabled, see the sidebar "**R5 templates with context-sensitive help**."

This article describes the new help features and how to use them to create context-sensitive help for your applications. It also includes some advanced ways in which to use the features. All of the features described in this article are showcased in a sample help system created for the Discussion template. This sample is available for **downloading** from the Sandbox on Notes.net.

Note: The **sample help system** for the discussion database should be used for demonstration purposes only. We did not attempt to write accurate help for the template; therefore, it should not be distributed to users. You must use R5 to run the sample databases, and all the databases must reside in the same directory.

Understanding the new help features

R5 includes three new context-sensitive help design features. Together, these features make it easy to create context-sensitive help for your users.

- An event -- HelpRequest -- for forms, views, folders, and pages
- An @command -- OpenHelpDocument
- A new Help window that stays on top and displays a customized menu

HelpRequest event

The programmer's pane includes a new event called *HelpRequest*. You can use this event with the following design elements: forms, views, folders, and pages. This event is triggered whenever users press F1 or choose Help - Context Help. As with other events, Notes executes the code that you write for the event whenever the event is triggered.

You can use the Notes formula language (@commands and @functions) to program code for the event. You cannot, however, use LotusScript or JavaScript to program code for this event.

Keep in mind that this event is not available for dialog boxes created with LotusScript or the formula language (such as, @Prompt or @DialogBox). So, if you have dialog boxes that appear in response to an action or button, you cannot provide context-sensitive help for these dialog boxes.

OpenHelpDocument command

To help you with writing code for the event, R5 includes a new @command called *OpenHelpDocument*. This @command lets you specify a help topic to display in response to an event.

Help window
The OpenHelpDocument command displays help in a separate, small window with a customized menu that provides only relevant help functionality for your users. For example, one of the menu choices, View - Always On Top, lets users display help and work in the application at the same time.

Before you start

The following are decisions that you have to make before you begin building context-sensitive help:

1. Where will your help topics be located?
You have two choices for locating help for a design element. You can either store topics inside the application or point to a database outside the application. Each of these methods has its advantages and disadvantages. You need to decide which meets the needs of your users.

If help is provided inside the application, you can distribute the application easily since you have only one database. However, if you are concerned about users making changes to help documents, you have the added task of setting up form security in order to allow users to have Editor access to the database and only Reader access to help documents.

If help is provided in a separate help database, you can display help in a separate, small window and users can set that window to always display on top. You can also set security at the database ACL level. The downside of using a separate help database however, is that you must have control over where the help database resides so the OpenHelpDocument formula points to the right database.

2. What type of design do you want for your help topics? For example, what will the body of the topics look like and do you want to include special views so users can browse help topics?
The help design can be as simple or as complicated as you want. We suggest, at a minimum, you should have:

- A form that contains a text field for the topic title, a rich text field for the topic content, and a hidden text field to use for displaying context-sensitive help using OpenHelpDocument. For the purposes of this article, we'll call the hidden text field the *Help ID* field, but this field can be named anything you want. We'll get to the details on the Help ID field later.
- A view for displaying all the help topics just in case your users want to browse for a specific topic or do a search.
- A view that displays all the help topics, listed in ascending or descending order by the Help ID field. This view is necessary for OpenHelpDocument to look up the topic. You can hide the view if you don't want users browsing it.

You might also want to use the design of the help databases that come with Notes and Domino. This template has been tested for problems and has been through numerous usability testing sessions, so you can be assured that your users will find it easy to use. In addition, using the system help templates provides consistency in the way that users obtain help. For information on how the R5 help design works, download the **Domino and Notes Release 5 Help Templates** posted on Notes.net.

3. Which forms, views, pages, and folders need context-sensitive help?

You can supply context-sensitive help for some or all of these design elements in the database. If you do not supply context-sensitive help for an element, Notes defaults to the system help. You and your users are the best

judges for determining which elements need context-sensitive help. You might want to do an informal survey of your users to determine which areas of the application are most problematic for them.

You should also decide whether you want to display different help depending on whether the user is editing or just reading a document or any other type of logic that you want to perform before you display help.

Make a list of the design elements for which you want to provide context-sensitive help. You're going to use this list later when you start writing the help topics.

4. How do you want the help topics organized?
It is always a good idea to create an outline before you start writing; however, this is not an article on technical writing, so we'll leave you to your own devices.

OpenHelpDocument: the syntax

Before you create help, you should be familiar with the @command *OpenHelpDocument*. This @command is similar to @DbLookup in that it looks up a document in a view using a key that you specify. OpenHelpDocument goes one step further: it opens the document found in the view.

The following is the syntax for the OpenHelpDocument formula:

@command([OpenHelpDocument]; *server*:*database*; *viewname*; *key*)

server is the name of the server where you put the help database. If this argument is a blank string or if you do not include this argument, Notes looks for help on the user's local hard drive. To use the name of the current server as the server name in the formula, use @Subset(@DbName; 1). This way, you don't need to know the name of the server on which the application resides, you simply need to make sure the application and help database are on the same server.

You do not need to provide the server argument if help is stored in the same database as the application.

database is the name of the database that contains the help topics. The database location is relative to the Notes data directory. If this argument is a blank string (""), Notes looks for help in the current database. If help is in a separate database and the database is not in the Notes data directory, you must supply the subdirectory name. To use the same subdirectory as the application, use @LeftBack(@Subset(@DbName;-1);"\\")

When you enter a string for the database name, make sure you use double backslashes (\\) if you are specifying a file in a subdirectory. For example, if the database is in c:\lotus\notes and the data directory is c:\lotus\notes\data, you specify the database like this: "c:\\lotus\\notes\\help.nsf"

server:*database* are two separate strings, so the colon (:) is not included in the quotes -- for example, "myserver":"helpme.nsf" not "myserver:helpme.nsf"

viewname is the name of the view that contains all the help topics. This view is hidden from the user and is sorted by a unique Help ID created for each topic.

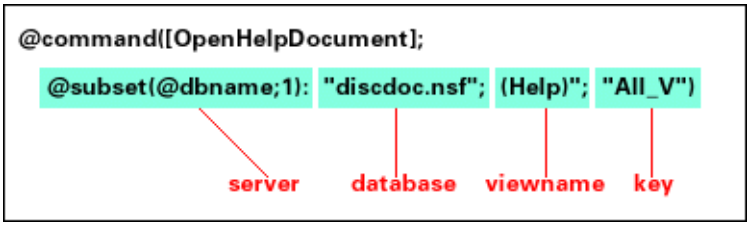
key is the unique Help ID that you provide in a field in each topic. Enter the Help ID for the topic you want to display for this design element.

All arguments are text strings; therefore if you enter literal text, it must be included in double quotes. Capitalization counts! (Except for the key

argument, which is case-insensitive.)

Examples
The following example opens the topic Main_F in the (Help) view of the current database when the user presses F1.

This next example opens the topic All_V in the (Help) view of the database discdoc.nsf, which resides on the same server as the application.



Creating context-sensitive help -- the basics

Now that your planning is complete -- you did create that outline, didn't you? -- and you understand the syntax for OpenHelpDocument, you are ready to get started. The steps below detail a version of context-sensitive help in its simplest form. Perhaps that's all you'll need for your application; however, if you find you need more complexity, there is an Advanced section below that discusses additional ways to create context-sensitive help.

1. If you are storing topics in a separate database, do the following:

- If you are using the system help design, download the **help design template** from Notes.net. Deselect the Database Property “List in Database Catalog” and remove the category name. You can then skip Steps 2 through 5 below.
- Create the database and set the ACL for users to Reader.

2. Create the form that you want to use for help. This form can contain any elements that you want; however, you must include a text field for the Help ID that the OpenHelpDocument will use to look up the topic. This field should be hidden from your users.

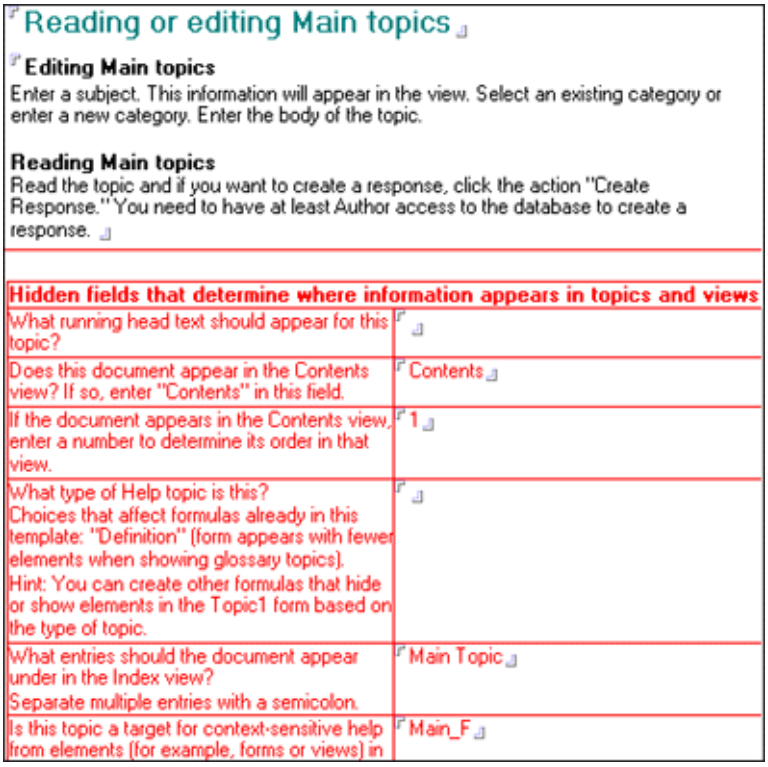
3. If you are storing topics inside the application and you don't want users making changes to your topics, create an Authors field on the form and specify the default hierarchical names of the users who can edit help topics -- for example, you and other designers. This field should only display when you edit a topic.

4. Create a hidden view that includes all topics that use the help form. If help is part of the application, then you need to enter a selection formula so only the help topics appear in the view (SELECT Form=”Help”). The Help ID text field must be in the first sorted column in the view (either in ascending or descending order).

5. Create any other design elements that you want your users to use when accessing help -- for example, a view that lists topics according to index entries or displays all topics for searching.

6. Using the help form and the list of elements you created that need

context-sensitive help, write help topics for each element. Enter a unique Help ID in the text field you created in Step 2, or if you are using the system help template, enter a Help ID in the AppSpecID field. This can be any text you want, but it's better if you use a format that's easy to remember -- for example, Main_F, which indicates the help topic for the form named “Main.”



Reading or editing Main topics

Editing Main topics
Enter a subject. This information will appear in the view. Select an existing category or enter a new category. Enter the body of the topic.

Reading Main topics
Read the topic and if you want to create a response, click the action "Create Response." You need to have at least Author access to the database to create a response.

Hidden fields that determine where information appears in topics and views	
What running head text should appear for this topic?	
Does this document appear in the Contents view? If so, enter "Contents" in this field.	Contents
If the document appears in the Contents view, enter a number to determine its order in that view.	1
What type of Help topic is this? Choices that affect formulas already in this template: "Definition" (form appears with fewer elements when showing glossary topics). Hint: You can create other formulas that hide or show elements in the Topic1 form based on the type of topic.	
What entries should the document appear under in the Index view? Separate multiple entries with a semicolon.	Main Topic
Is this topic a target for context-sensitive help from elements (for example, forms or views) in	Main_F

7. Write any other topics that you want to link to the context-sensitive help topics -- for example, conceptual information. Create the links from the context-sensitive help topics to the conceptual topics. These topics do not need a Help ID.

8. In the application, open each form, view, page, or folder for which you want to display context-sensitive help.

9. In the HelpRequest event, enter the OpenHelpDocument formula (see the previous section of this article for the syntax.) Use the unique Help ID for the key argument.

10. Save the form, view, page, or folder.

11. Test to make sure help appears correctly by opening the form, view, page, or folder in the application and pressing F1.

That's all there is to it! Seems easy, doesn't it?

Advanced topics

Now that you understand the simplest way to create context-sensitive help, let's talk about more complex context-sensitive help features.

As mentioned above, there is nothing special about the HelpRequest event. It works just like any event by executing code that you write. The only limitations are that it uses only the Notes formula language (no LotusScript or JavaScript) and does not work for forms displayed in dialog boxes, for

example with @DialogBox. Other than that, you can design a complex help system that uses any of the features supported by the Domino Designer.

The following sections provide some ideas for more complex help systems. You can use them as is or modify them to meet the needs of your organization. These features are also included in the sample database.

Creating field-level help
How many times have you been frustrated by an application when trying to figure out what to enter in a field? You press F1 and the help system tells you how to fill out the form. You know how to fill out the form, but what the heck are the parameters for the xyz field?

If you want, you can use JavaScript to create field-level help; however, you cannot use JavaScript to respond to a HelpRequest event (that is, when the user presses F1). For information on using JavaScript to create field-level help, see the article "**Creating field help for your Domino applications**".

If you want to display field-level help whenever a user presses F1 in an application, you can use both the field Entering event in LotusScript and the HelpRequest event for the form to provide application context-sensitive help. Here's how to do it:

1. Create a hidden text field on the form the user is filling out and name it something like *Current_field*. This field keeps track of the current location of the cursor.
2. For each field that you want to display context-sensitive help, enter the following LotusScript code for the Entering event:

```
Dim workspace as New NotesUIWorkspace
Dim uidoc As NotesUIDocument
Dim doc As Notesdocument
Dim item As NotesItem
‘Get the current document
Set uidoc=workspace.CurrentDocument
Set doc=uidoc.document
‘Set the Current_field field to a value that indicates where the cursor is located. Displayed in bold below
Set item=doc.ReplaceItemValue(“Current_field”, “Body”)
```

3. Create help topics for each of the fields and in the Help ID field, enter the name you used to specify where the cursor is located (displayed in bold above).

4. On the OpenHelpDocument command for the form, specify the contents of Current_field for the key in the OpenHelpDocument formula, for example:

```
@command([OpenHelpDocument]; “”; "(Help)"; Current_field)
```

5. In the Exiting event for the field, reset Current_field to blank -- that is, replace “Body” above with “”.

You can see an example of this in the sample Discussion database when you fill out an Author Profile.

Displaying a list of topics for a design element
If you want to give users a choice of topics when they are using a form, view, page, or folder, simply use @Prompt to display the list of choices when the user presses F1. Then, after the user makes a choice, use OpenHelpDocument to open the document.

Here's an example of the code that you can use to accomplish this. Add the code to the HelpRequest event for the form, view, page, or folder.

```
openfirst := “Response_New_F”;
opensecond := “Response_Edit_F”;
openthird := “Response_Read_F”;
REM “Show users a list of the selections.”;
choice := @Prompt([OKCANCELLIST]:[NoSort];”Select a Topic”;”Select a
topic to display”;”Entering a new response”;”Entering a new
response”:”Editing an existing response”:”Reading a response”);
REM “Figure out which item the user selected and set the variable opendoc to
the selection”;
opendoc := @if(choice=”Entering a new response”;openfirst;choice=”Editing a
new response”; opensecond; choice=”Reading a response”;openthird;@false);
REM “Open the help topic the user chose based on the value of opendoc”;
@command([OpenHelpDocument];@Subset(@Dbname;1):”discdoc.nsf”;
”(Help)”;opendoc);
```

You can see an example of this in the sample Discussion database when you read or edit a Response document.

Creating context-sensitive help for Web applications
Since the browser does not capture when users press F1 in an application, you must program a button or action that displays help for Web users. If your application serves a dual-purpose for Web users and Notes users, you will want to hide the button or action from Notes users.

Creating context-sensitive help for Web applications is as easy as programming that button or action using the OpenHelpDocument command. Here's an example of the Main topic form that includes a Help action button:

In the simplest version, the code for the button is:

```
@command([OpenHelpDocument];@Subset(@DbName;1):”discdoc.nsf”;
”(Help)”;”Main_F”)
```

However, you can use any other formulas supported by the Web server, for example, if you wanted to display different help depending on whether the Web user was editing or reading a document, you can use @IsDocBeingEdited.

The Web server opens the help document you specify. It does not, however, open help in a separate window. You can see an example of this in the sample Discussion database when you read a Main topic with a browser.

Using additional keywords for searching
As you probably already know, full-text search displays topics that contain a word in the document. This makes it difficult for users to search for synonyms. For example, if you use the word “personnel” in your help topics and users use the word “staff” when searching for those documents, they won't find the documents.

One way to correct this problem is to add synonyms to a hidden field in the document. When a user searches for the synonym, Notes displays the topics that contain the synonym in the hidden field.

We added a field named "Synonyms" to the Topic1 form in the sample help database. This field is not available in the system help template that you can download from the Web.

To see this work, create a full-text index of the database and search on the word "reply," which cannot be found in the visible fields in the template. This search yields hits because the Synonyms field contains the word "reply."

Conclusion

R5 lets you create context-sensitive help for your applications using the Domino formula language. The help that you provide can be as complicated or as simple as you want -- your only limitations are those that already exist in the formula language. So get creative, and then share your designs with others in the **Iris Sandbox** and/or the **Notes/Domino discussion forum** on Notes.net.

ABOUT THE AUTHOR

Jo-Ann Spera is a principal writer in the Domino/Notes UA group and has worked at Lotus for 9 years. She worked on security documentation and participated in the design of the help system for R5.

[back to "Creating context-sensitive help for applications"]

R5 templates with context-sensitive help (sidebar)

The following is a list of R5 templates that supply context-sensitive help. These templates display help in either the Notes 5 Client Help (help5_client.nsf) or Domino 5 Administration Help (help5_admin.nsf) database. You can override the help provided by changing the formula for the HelpRequest event in the design elements.

Personal Address Book (pernames.ntf)
Domino Directory (pubnames.ntf)
Domino Certificate Authority (cca50.ntf)
Server Certificate Admin (csrv50.ntf)
NNTP Discussion (R5.0) (nntpdi50.ntf)
Statistics and Events (events4.ntf)
Statistics Reporting (statrep5.ntf)
Mail (mail50.ntf)
Bookmarks (bookmark.ntf)
TeamRoom (5.0) (teamrm50.ntf)

Lotus Home | IBM Home | Iris Home | Feedback