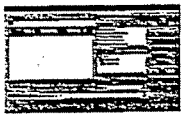
Excel 95 spec

Word Perfect File Open Dialog



Things in WP file open that isn't in the Chicago file open dialog.

- Directory Quick lists - listbox for selecting and a button for editing the list and how it appears in the dialog
- "Commands" button with menu. This is available through context menus in Chicago's file open which has the same functionality but is less discoverable
- View window - if Chicago follows through with its plan to cut this feature.
- Larger file open window - this is not a clear advantage with the new commdlg view pane.
- Extended file info, not relating to a specific file. WP's dialog shows you stuff like the total size for a group of selected files, and the amount of free disk space. Nice, but not essential for us to do.
- Ability to set the default directory and file extension. I just got a report from PSS that this is a sizable call generator for us, especially the latter.

HIGHLY CONFIDENTIAL

M CONFIDENTIAL

MS 0171040
CONFIDENTIAL

FL AG 0067420
CONFIDENTIAL

Shared Office Location for Templates and Wizards

Program Mgr Russell Williams

Summary

With EMS it will be very easy to share information in a workgroup. Files and information used by Excel and the other Office applications such as templates, Wizards, and AutoQueries should be equally easy to share. This feature defines a common place for storing and accessing such information.

HIGHLY CONFIDENTIAL

M CONFIDENTIAL

MS 0171041
CONFIDENTIAL

FL AG 0067421
CONFIDENTIAL

Justification

EMS will be promoted as a friendly way to share and exchange information in the Office '95 timeframe. Information other than user documents should be sharable via this mechanism.

Description

On EMS the feature consists of a Public folder of known name that is used to store shared files. This feature operates on a file level, not a property level. Each shared file corresponds to exactly one EMS message. The message can be either a transparent envelope or a regular message with an attachment (for our purposes it doesn't matter).

Categories of information are accessed by using fixed folder names below the root shared folder. In the case of Excel Auto Queries, the folder tree might look like:

Microsoft Office
Microsoft Excel
AutoQueries

The Microsoft Office folder is specified as the root shared folder by an administration UI, TBD. Applications such as Excel use a layer to iterate the files contained within a named folder of the shared folder tree, for example the "Microsoft Excel:AutoQueries" folder. On EMS the layer simply iterates the messages in the folder using MAPI. If the application wishes to use the file the layer copies it to the user's hard drive into a temporary location (MAPI should do this automatically).

The EMS shared folder location is be automatically detected by a client copy of Excel based on the presense of MAPI on the system.

How to put stuff in? Use the Capone client and simply drag/drop the file into the desired folder. Capone/MAPI automatically copies the file into the mail store.

This feature operates like a vastly simplified version of InfoCenter. Note that this feature is **not** trying to be a file system or a document store, only an reference information store for known information used by Excel. We're using MAPI simply as a friendly network. All file manipulation the client side still goes through DOS (via the temp file shuffle mentioned above).

The location should also be accessible to FAT users. Here we'd have to define some INI switch on each client machine to specify the right server (UNC path) containing the shared files.

HIGHLY CONFIDENTIAL

M CONFIDENTIAL

MS 0171042
CONFIDENTIAL

FL AG 0067422
CONFIDENTIAL

Case 2:04-cv-01045-JFM Document 502-12 Filed 03/09/12 Page 4 of 83

Platform

This could be difficult on the Mac depending on the level of MAPI support supplied by WGA on the Mac. Not a problem for the '95 releases.

International

The name of the shared folders should be localized. In mixed-language environments this could cause a problem since the apps will probably bind to the folder by name. Possible solutions are:

- Use a ID-binding system on MAPI, where the ID of the root shared folder is communicated to the clients via a MAPI property in a fixed location
- Try the localized foldername first, and if that fails then fallback to the English name.

Visual Basic

Property to determine where the shared location is.

Access to the shared location via VBA - a layer onto the shared file access layer used by Excel. Maybe just DLL calls?

User Ed

The shared location should documented once for all Office apps and reused appropriately.

Changes

5-Apr-94	First Spec	russellw

HIGHLY CONFIDENTIAL

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF KANSAS

IN RE URETHANE ANTITRUST LITIGATION	MDL 04-1616 (JWL/JPO)
	Civil Action Nos. 08-2617, 09-2026
THIS DOCUMENT RELATES TO: POLYETHER POLYOL CASES	

JOINT MOTION FOR APPROVAL OF BRIEFING SCHEDULE FOR RESPONSES TO DIRECT ACTION PLAINTIFFS' FIRST AMENDED COMPLAINTS

The Parties hereto, through their respective counsel of record, hereby jointly move the Court for an order approving and setting the following schedule regarding Defendants BASF SE, BASF Corporation, The Dow Chemical Company, and Huntsman International LLC's ("Defendants")[1] responses to the First Amended Complaint filed on March 19, 2009, by Direct Action Plaintiffs Carpenter Co. and other companies listed in full in the signature block below, in Carpenter Co., et al. v. BASF SE, et al., Civil Action No. 08-2617 (D. N.J. No. 08-cv-5169) ("Carpenter Direct Action"), and to the First Amended Complaint filed on March 19, 2009 by Woodbridge Foam Corporation and related companies listed in full in the signature block below, in Woodbridge Foam Corporation, et al. v. BASF SE, et al., Civil Action No. 09-2026 (D. N.J. No. 08-cv-5759) ("Woodbridge Direct Action", collectively "the Urethanes Direct Actions"). In support of this motion, the Parties hereto state:

1. On October 17, 2008, Carpenter Co. and a number of other Plaintiffs filed the Carpenter Direct Action Complaint in the U.S. District Court for the District of New Jersey. On November 20, 2008, the Judicial Panel on Multidistrict Litigation ("Multidistrict Panel") entered

[1] Subsequent to the filing of Direct Action Plaintiffs' initial Complaints, on or around January 6, 2009, Defendant Lyondell Chemical Company filed for bankruptcy. As a result, the Urethanes Direct Actions are currently stayed as to Lyondell.

9. Direct Action Plaintiffs seek an order setting May 28, 2009, as the deadline for Direct Action Plaintiffs to file their responses to Defendants' motion(s).

10. Defendants seek an order setting June 29, 2009, as the deadline for Defendants to file their replies to Direct Action Plaintiffs' responses to Defendants' motion(s).

11. Defendants BASF Coordination Center Comm. V., Jean-Pierre Dhanis, and Uwe Hartwig were added as Defendants in Direct Action Plaintiffs' First Amended Complaints. Because they have not yet been served with the First Amended Complaints, they are not included in this Motion. Accordingly, at this time, the Parties do not seek by this Joint Motion for any schedule to be set for these Defendants' responses to the First Amended Direct Action Complaints.

WHEREFORE, the Parties hereto respectfully request that the Court enter an Order granting the Parties' Joint Motion For Approval of Briefing Schedule For Responses to Plaintiffs' First Amended Complaints in the Urethanes Direct Actions.

Dated: April 8, 2009

/s/ Jodi Trulove
Jodi Trulove
Richard J. Leveridge
Elaine Metlin
DICKSTEIN SHAPIRO LLP
1825 Eye Street, NW
Washington, DC 20006
Tel: (202) 420-2200
Fax: (202) 420-2201

/s/ David F. Oliver
David F. Oliver KS #70731
BERKOWITZ OLIVER WILLIAMS
SHAW & EISENBRANDT LLP
2600 Grand Boulevard, Suite 1200
Kansas City, MO 64108
Tel: (816) 561-7007
Fax: (816) 561-1888

Europe S.A.S., Tramico S.A.S., UAB Vita Baltic International, Veenendaal Schaumstoffwerk GmbH, Vita Cellular Foams (UK) Limited, Vita Industrial (UK) Limited, and Vita Polymers Poland Sp. Z.o.o.

/s/ Jodi Trulove

Jodi Trulove
Richard J. Leveridge
Elaine Metlin
DICKSTEIN SHAPIRO LLP
1825 Eye Street, NW
Washington, DC 20006
Tel: (202) 420-2200
Fax: (202) 420-2201

R. Bruce Holcomb
Christopher Leonardo
ADAMS HOLCOMB LLP
1875 Eye Street, N.W.
Suite 810
Washington, DC 20006
Tel: (202) 580-8820
Fax: (202) 580-8821

Counsel for Plaintiffs Woodbridge Foam Corporation, Woodbridge Holdings, Inc., Woodbridge Corporation, Woodbridge Foam Fabricating, Inc., Woodbridge Services Inc., Hairlok Limited, Woodbridge Foam Deutschland GMBH, and Woodbridge Foam (UK) Limited

Justin T. Toth
John W. Mackay
36 South State Street, Suite 1400
Salt Lake City, UT 84111
Tel: (801) 532-1500
Fax: (801) 532-7543

Harry M. Reasoner
David T. Harvin
Lauren Harrison
First City Tower
1001 Fannin Street, Suite 2500
Houston, TX 77002-6760

Counsel for Huntsman International LLC

OLE Property Exchange

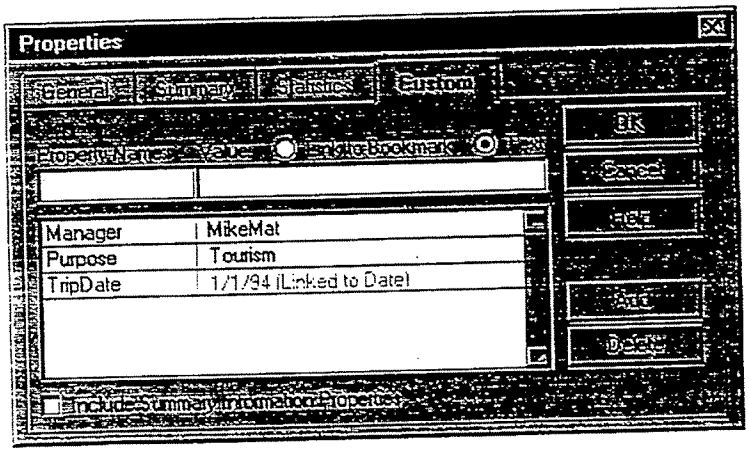
Program Mgr Heikki Kanerva[1]

Summary

The Office 95 applications will support the ability to create user-defined properties. These properties are stored in a standard way so that they can be *promoted* for use in document containers such as EMS/Capone, Ren, OFS, or even Chicago property sheets, without having to launch the app that created the document. This spec describes exactly how the information will be stored and used in the OLE compound files created by the Office apps, so that the information can be shared with other document containers in a standardized way. It also describes the new user interface for editing standard properties and creating and editing user-defined properties. User-defined properties may be simple values, or links to document content.

This spec also describes what we believe are the major clients for using these properties, and how the properties will be used by each client. The primary functionality that will result from this spec is that a user will be able to drop a document (created by any Office app) into a Public Folder. The properties that the document exposes — both the standard and the user-defined properties — will become available for use as a column in a Capone custom view.

Below is the dialog for setting user-defined properties in the apps.



HIGHLY CONFIDENTIAL

[1]This spec is based on one written originally by Mike Mathieu. These additional people have contributed heavily to this spec at various times: Russell Williams, Richard Greene, Keith Grochow, David Tuniman, and Heikki Kanerva.

Justification

Called "OLE Property Exchange," this open, publicly available technology allows us to link the unstructured world of document content, with the process-oriented world of business systems and information exchange. This makes the technology a crucial integration point between our Office applications and EMS. It competes directly with Lotus Notes Field Exchange (Notes/FX) and should be evangelized by DRG as an important step for a document- or content- oriented application to tightly integrate with Microsoft Information Exchange.

The resulting user benefits include the capability to browse and search using their user-defined properties. Custom views will also provide for a simple reporting mechanism for a document database.

Description

The Concept

OLE Property Exchange allows unstructured content (created by Office apps) to serve as data for more structured, database-focused containers. The result is typically some sort of *document database* which gives users some of the benefits of a relational database, without some of the structured overhead, all while working with documents. These databases may or may not have associated code which can turn them into a "workgroup application."

An application which saves its files in OLE 2.0 compound file format (a.k.a. docfiles), and also follows the standard for exposing its custom and standard properties, will be able to seamlessly exchange their property values with Property Exchange-aware containers, such as Capone, Ren, Elmer, and the Chicago property browser. This will all work without the container having to run the app which created the document.

Property Sets

Summary Information Properties

Appendix B of the OLE 2.0 SDK defines the standard data format for OLE properties.[2] In particular, Office 4.x apps already support the "\005SummaryInformation" named property set. Each of the eighteen defined properties in this property set will be available for OLE Property Exchange with document containers. The table of OLE standard properties can be found in "Appendix A – Standard Document Summary Information" in the end of this document.

MSOffice Summary Information Properties

Office95 apps will also support a new property set called "\005MSOfficeSummaryInformation" which will provide the enhanced properties, over

[2] Unfortunately, no APIs are defined or supported. OFS will define OLE interfaces for accessing property storage. DavidStu is a contact if you're interested. Note that the OLE sample code also contains some examples of code to read/write properties.

HIGHLY CONFIDENTIAL

and above those found in the SummaryInformation property set. For a list of new properties, FMTID etc. see "Appendix B -- MS Office Summary Information".

ISSUE: I still need to check to see if there are properties that we're including Notes/FX support in 1994 that we need to add properties for in 1995 OLE Property Exchange support. Our goal is to have OLE Property Exchange be a complete functional superset of our Notes/FX implementation. KeithG is the contact person.

ISSUE: What type are we going to use to support multi-valued string fields? What is MAPI doing? Can Capone support them in a view? JonKauf is the MAPI contact.

ISSUE: How do we want to handle a generic equivalent of PowerPoint's Notes/FX LaunchSlideShow property? Should this just be PowerPoint? KeithG is the Notes/FX contact.

User-Defined Properties

A third property set, "\005UserDefinedProperties," will contain string-named properties, rather than using property ids. This will allow users to create any property they want with any name they want. The only difference between this property set and the other property sets, is that properties in this property set will require a name for each property, using a property set dictionary. The OLE Programmer's Reference indicates that names are optional, but they will be required for OLE Property Exchange support of user-defined properties.

The FMTID for the UserDefinedProperties property set is:

D5CDD505-2E9C-101B-9397-08002B2CF9AE

Use the DEFINE_GUID macro to define the FMTID for the property set.
DEFINE_GUID(FormatID_UserDefindProperties, 0xD5CDD505L, 0x2E9C, 0x101B, 0x93, 0x97, 0x08, 0x00, 0x2B, 0x2C, 0xF9, 0xAE);

Hiding Property Names

We will use a simple naming convention to allow users to indicate that they do not want a property name to show up in any container UI that allows users to see the names of properties (see the later section on column-picking). For instance, a backup utility might create a custom property on a document called "_ _NortonBackupLastBackDateTime" used to help it optimize its backup procedures. But, a user would never want to see this property name in Capone when creating a custom view.

To solve this problem, we use the simple rule that if the property name begins with a # sign, then the name will be hidden.

ISSUE: Is there some better symbol to use than "#"? Other possibilities are @, !, *, %, $, or _. Parser issues in Excel or Word? RussellW is the Excel contact. Rgreene is the Word contact.

HIGHLY CONFIDENTIAL

NOTE: this convention is not enforced at the property level. Rather it is required that document containers support this convention by simply not showing the #properties in any lists of properties on the document presented to the user. However, the applications or property editors which allow the user to create such properties should show these hidden properties in their lists, so that the users can delete them if they want.

Name Collisions

When a user-defined property and a property in SummaryInformation or MSOfficeSummaryInformation conflict (they have the same name), then the user-defined property should be used, rather than the standard property. This allows for a scenario where the user wants to replace a built-in property with his own, possibly of a different type.

It is up to the individual container to decide whether or not they want to provide a user interface for selecting or resolving which property to use. An arbitrary naming convention would then be used to distinguish the two. For instance, if there was the SummaryInformation property Author, and a user-defined property named Author for a document, and it was dropped into a Capone folder, then when the user went to go create a custom view, the listbox would show all of the fields, including something like "Author" and "User:Author" to distinguish the user-defined property from the standard one. Again, the only hard and fast rule here is that if there is no other means provided to distinguish between like-named properties, then the user-defined property wins out.

Capone and Ren Integration Issues

When the user drags a file from the file system and drops it in a Capone folder, Capone initiates a process to read out the properties from the document, and promote them to be MAPI properties.

The details of what happens are explained in the Capone spec in section 9.2.

There are three different sets of properties that the container (Capone) needs to promote. They are:

- The 18 standard SummaryInformation properties
- The new standard MSOfficeSummaryInformation
- All of the UserDefinedProperties

Once the properties are promoted, they become available in the Group, Sort, Filter, and Views UI.

In all of these cases, any user-defined names formed as described above in the section on "Hiding Property Names" should not be displayed in the list of property names anywhere in the UI.

Office 95 Application Support

Word, Excel, and PowerPoint will support OLE Property Exchange in their 1995 releases. This means that each application will support the SummaryInformation, MSOfficeSummaryInformation, and UserDefinedProperties property sets. Each

HIGHLY CONFIDENTIAL

application will also provide a consistent File Properties dialog to replace the existing Summary Info dialog.

Unlike in today's versions of the apps, each app will store the property information *only* in the OLE property sets. There will be no duplication of the data in the apps' native data. This will reduce the change of the property set information getting out of sync with the native data.

Word needs to save a thumbnail property value in the SummaryInformation property set, which it currently does not do in Word 95.

When a document is loaded:

- The contents of the custom properties are read off the property stream and stuffed into the source of linked properties (names, bookmarks).
- If the property stream contains additional properties beyond those defined within the document (i.e. properties added by an external application that rewrote our property stream), we will just use them as static-value properties, or ignore them if they are in their own property set that we do not recognize.
- In Excel, if the source cell of the link contains a formula we ignore the incoming property value and leave the formula alone. In Word, if the source of the link is a field we ignore the incoming property value and leave the field alone. The exception to this is field which can have user input results. The key example being form fields. This will allow us to have two-way links for form fields, without blowing away any field in general.

When a document is saved, the document properties are evaluated and pulled from the source of any linked bookmarks or cells. Any circular references are simply evaluated once and then stored. Static properties require no evaluation.

Special Cases in Excel

When evaluating linked ranges in Excel, if the range is larger than a single cell we use the upper-left cell of the first area of the range. When writing out properties, if a name evaluates to an array, we pick the upper-left element.

Special Cases in Word

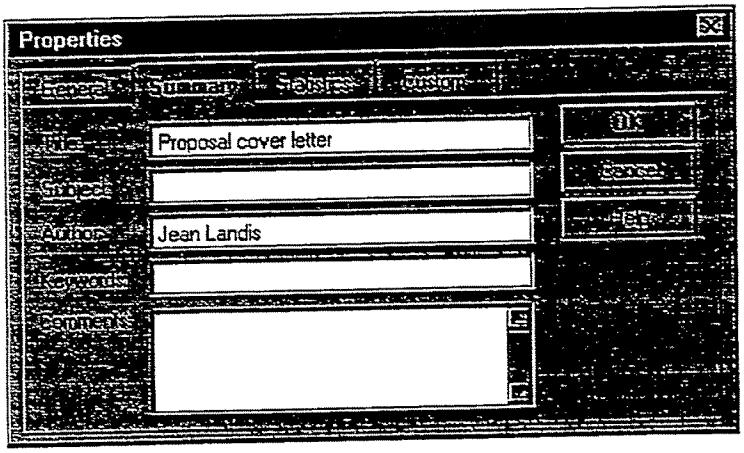
Bookmarks can refer to some special objects inside Word which do not readily fit into standard properties.

We need to think about how we want to handle these cases (and if these are all of the cases). The thing that we'll likely do is just remove them from the list of available bookmarks to which you can link a property.

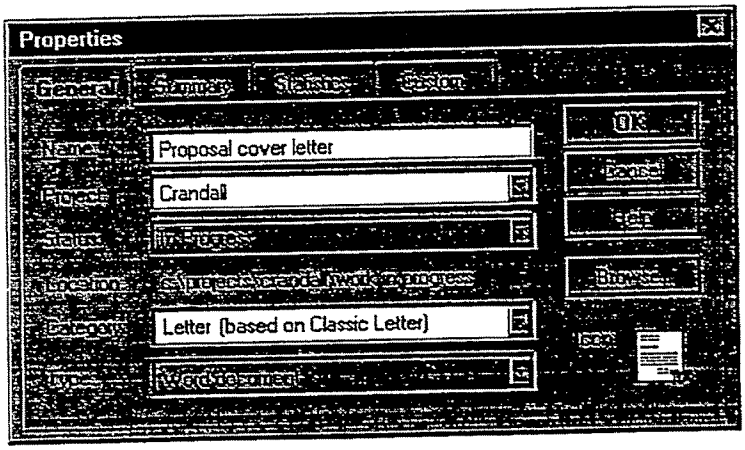
- Bookmark across multiple cells in a table.
- Tables in general.
- Pictures
- OLE objects
- Frames

HIGHLY CONFIDENTIAL

Which tab as default?
We might want to have General tab be the default too, for consistency, but the fact is that most users will go to this dialog for the summary info functionality that it provided in Office 4.x.



These fields are the same as those supported in today's Summary Info dialog. The General tab provides more general information about the document.

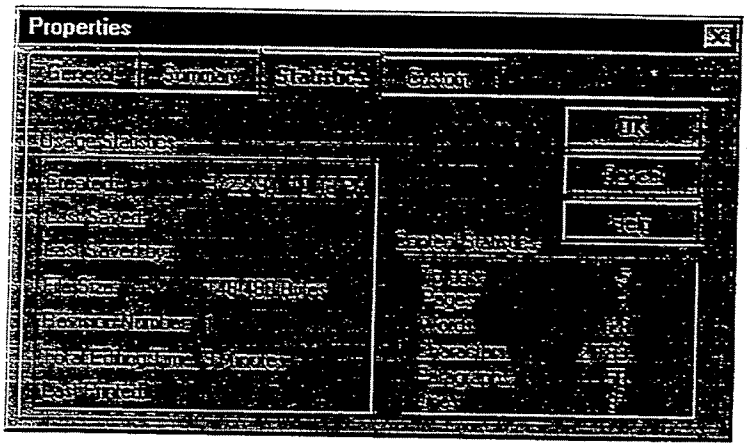


NOTE: This dialog is not yet finalized. We still need to work out if it's possible to support the multi-valued fields like Project or Status. This current dialog also makes it possible to use this dialog virtually as a replacement of Save As.[4] It's not clear that we will adopt this aggressive stance for Office 95.

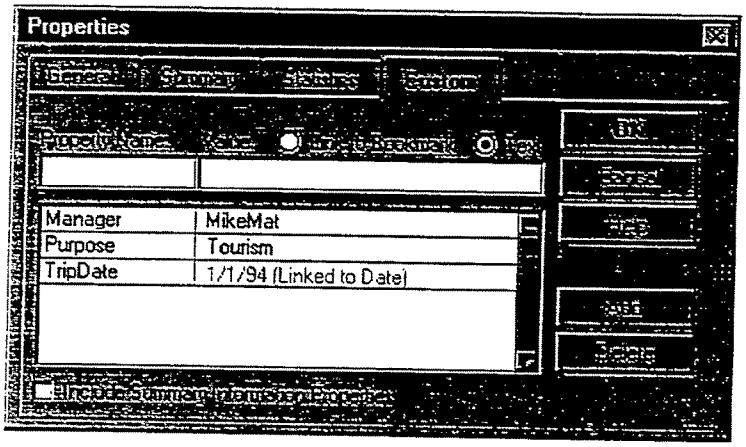
The document Statistics tab presents pretty much the same information as in Office 4.x, and it looks something like:

[4] This is from a proposal by DavidTu.

HIGHLY CONFIDENTIAL



Finally, the Custom tab provides a new interface for users to create user-defined properties that have either a text value, or are linked to some document content, such as a Word bookmark, or an Excel named range.



The checkbox for **Include Summary Information Properties** adds the editable properties from the Summary and General tabs to the listbox, so that the user can edit all of their properties in one place, if desired. These pre-defined properties cannot be deleted, but they can be linked to bookmarks or cells.

The mechanism here for adding new properties and modifying their contents is very similar to Word's AutoCorrect dialog. The user types a new property name and a value for that property in the edit controls at the top of the dialog, and then presses the **Add** button to add the property and its value to the listbox below. In Word, if the user chooses **Link to Bookmark**, the edit control changes to a dropdown list which contains all the bookmarks currently defined for that document.

In Excel the option is **Link to Name** instead of Bookmark. When picked, the Value editbox is a combo dropdown that is filled with the currently defined range names for the workbook, including local names. The user is allowed to type in a hidden name, assuming it already exists. Hidden names are not shown in the name list.

HIGHLY CONFIDENTIAL

When a property is linked to a bookmark or range, the linked value shows up followed by "(Linked to *bookmarkname*)".

Selecting a property in the list box causes its name and value to appear in the edit controls above (just like AutoCorrect). In this case, the **Add** button changes to **Modify**, but if the user changes the property name, the button reverts to **Add**. The **Delete** button deletes a property from the list, with the warning "Are you sure you want to remove the property 'foobar'?" <OK> <Cancel> <Help>.

ISSUE: Is the comment in the following paragraph still true? How does this mesh with the Notes/FX stuff, or the hidden property stuff described above?

We will continue to support the convention of publishing all bookmarks and names that have a PROP_ prefix as properties. These properties will be mapped to the name of the bookmark or name minus the prefix. i.e., a bookmark called PROP_SUMMARY will be published as the property SUMMARY.

Excel Issues:

1. Can we support names that are defined to values or formulas? Intuitively we should be able to evaluate any name in a similar manner regardless of what it is defined to be. If not supported, the dropdown list of names does not contain names that aren't defined to be references.
2. It would also be nice to have the combo-box be a refedit to allow the user to type in or semi-select a raw cell reference without having to first define a range name. According to our SDM guru, a combo-edit can theoretically be a refedit without much trouble, and such a beast was created for Data Sort in Excel5 but was cut before ship. If not, we should still allow type in references.

   We could also go with a standard refedit and no dropdown. The refedit would allow semi-selection of cells and enables PasteNames (F3) to paste in a name from a pick list. The paste-names behavior is less obvious than a dropdown of names, but it works today.
3. Excel currently doesn't have the equivalent of the fancy lines and grayed text in a listbox. If this is expensive for Excel we should punt.

Word Fields and Excel Formulas to Refer to Properties

Word will add a new field called DocProperty which takes a single parameter — a quoted string of the name of the property whose value you want returned. This complements today's built in fields for Summary Information like Author and Subject. This field will be listed in the Document Information category.

Excel will add a new worksheet function called DocProperty(*PropName*) which returns the value of the property *PropName*. The function should recalc when the document is opened, or when any property is changed from within the app, but it does not have to respond to changes to the docfile properties by an external application while the document is open.

Security

There are scenarios where the document properties themselves can contain sensitive information. So, when a file is password-protected, it might be desirable to also

CONFIDENTIAL
M CONFIDENTIAL
MS 0171052
CONFIDENTIAL
78
FL AG 0067432
CONFIDENTIAL

protect/encrypt the property values, or just not write them out at all. Most of the ways to attack this problem are either not secure (they provide a false sense of security, such as the PID_SECURITY value), or they are expensive to implement (provide encryption access to the individual properties outside of the app native data.)

Therefore, the current plan of record is to simply rely on the security of the container to provide robust data security services, and to do nothing special in the applications.

ISSUE: One other simple option is to simply not write out any property information when the file is password-protected. The problem then becomes where do we store the property values? We're assuming that these values are only stored in property sets. Another possibility is to provide a checkbox option that would let the user decide whether or not they would expose the properties in the docfile streams. However, this is bad because it leaves the security of the document open to the chance that the user forgot or simply did not know about the checkbox.

Chicago Integration Issues

Ideally this File Properties dialog used in the applications will be identical to (and share the code with) the extensible property browser handler in Chicago. The only difference is that under Chicago, the user would probably not be able to add new linked properties, or modify properties that were linked, other than to unlink them entirely.

Impact on Find File

The Find File 2.0 shared library provides the ability to browse document properties through a standard interface. The work required to read these properties out of the docfiles is the same work required to browse the properties in Chicago and should be shared.

Bound Control for Property Access

EricZ in the Access group has proposed that a custom control be written which would allow app writers to easily read and write properties using the three standard property sets. Contact Eric for more details on the progress on this. It too could share the code with the File Properties, Chicago Properties, and Find File properties features.

Notes Field Exchange Support

In Office 95, we will use the same method for registering with Notes and getting and setting Notes fields as we do in the Notes F/X feature in Office 94.5. But we will still carry out reading and writing properties to our property stream at the same time. This will ensure that our properties are always in sync with the document if it is moved out of the Notes database. The apps will:

- Read from their property stream and then read from the Notes database when launched from Notes.
- Write to their property stream and then the Notes database when updating Notes.

CONFIDENTIAL

ISSUE: Notes version 4 will be out in late '94 or early '95. They are promising improved support for F/X in the form of OLE2. We need to make sure we support the OLE2 form of F/X if there are any changes.

ISSUE: Are there any fields that we're including in Notes/FX support in 94 that we've left out?

Platform

This stuff should all just work going forward with Cairo and OFS. (Please speak up if it won't.)

Mac support should be identical to Windows for the feature dialog and OLE properties.

Notes F/X support only exists on 16-bit Windows. Notes does not yet exist for 32-bit Windows, and there is no API support for the Notes Mac client. We'll need to track the delivery and availability of Notes 4.0 to see if we'll be able to do anything in 1995.

Localization

User-defined property names are never localized. The names that we use in the user interface for a given property ID should be localized, just like we do today for Summary Information. This is one of the benefits that a propid provides — language independence. Note that this means that it will be difficult for EMS app writers to localize their views so that the same view shows up localized for different users.

Visual Basic

This will need to have extensive VBA and WordBasic support. A nice goal is to allow users to replace the File Properties dialog with a custom form. The Add and Delete buttons in the Properties dialog will require new WordBasic commands and VBA methods.

VBA Support: Excel

Excel already supports the standard summary info fields in the form of the Author, Subject, Keywords, Title, and Comments properties of the Workbook object.

```
ActiveWorkbook.Title = "Extensible Document Properties"
```

To support custom properties, we add the following:

DocumentProperties Collection

A standard collection object that contains all properties on a given workbook. For future compatibility, it contains both built-in standard properties and custom, user-defined properties. It supports:

Count property	Number of properties defined, as Integer.
_NewEnum method	Standard iterator to enable For...Each
Parent property	The Workbook object containing the properties

CONFIDENTIAL

[redacted]

Add(name, type, definition) method	Adds a new custom property. The **name** parameter is text, the **type** parameter is one of xlStaticValue or xlLinkedRange, and the **definition** parameter is either the static value, a Name object, a Range object, or the string rep of a Name or Range. In the future we may allow a **type** of xlBuiltIn, which adds back a previously deleted built-in property such as "Page Count". Adding a property that already exists replaces the existing definition unless that property happens to be built-in, in which case an error occurs. Returns the newly created DocumentProperty object.

Document properties are accessed using the **Workbook::DocumentProperties(index)** method, which when used without an argument returns the entire DocumentProperties collection object, and when used with an index returns a particular DocumentProperty. The index can be either the text name of the property or a numeric value from 1 to DocumentProperties.Count.

DocumentProperty Object

Represents a single document property definition. DocumentProperty objects are contained within the DocumentProperties collection. A DocumentProperty can represent either a built-in property such as "Title" or a custom property. We allow built-in properties to create a consistent access mechanism and for future compatibility, where we may allow the user to delete/add selected built-in document properties to the list of exposed properties.

Type property	One of xlStaticValue, xlLinkedRange indicating whether the object is linked.
Name property	Name of the property, as text.
Parent property	The Workbook object containing the property.
Delete method	Deletes the property. Deleting built-in properties is not allowed in Excel '95.
Definition property	The definition of the property, either a text value or a Name or Range object. For setting, we allow a string to be used in place of a Name or Range object (since Excel doesn't support PropertySet very well). We avoid using "Value" for this property because it has the unwelcome side-effect of becoming the default-value for the object. Also, if we ever want to add support for programmatically evaluating a property before saving the document, "Value" is the obvious

Filed in CONFIDENTIAL

M CONFIDENTIAL

MS 0171055
CONFIDENTIAL

FL AG 0067435
CONFIDENTIAL

"choice.

BuiltIn property	Indicates whether the property is a standard property built-in to Excel by returning True for built-in and False for user-defined. A programmer can use this property to discriminate user-defined properties when iterating all the properties of a document.

In the future we might support ordering of properties and allow instant evaluation.

This model does not account very well for non-exposed document properties that lie dormant until they're added by the user. The primary problem is that the user might want to evaluate such a property without having it be exposed. Should this problem arise in the future we can add direct Workbook object properties if a built-in property is truly that common. For example, we could add a Workbook::PageCount property and not bother going through the property collections. Keeping the property collections clean costs us no functionality in the short term, and in the long term will expose us to less code breakage as a result of poorly written addins suddenly having to deal with new built-in dormant properties. A secondary problem with this model is that it doesn't handle iteration of dormant properties. This can be worked around in the future by adding a top level property that returns an array of the names of all possible built-in properties, e.g. Application::BuiltInDocumentProperties => { "Page Count", "Last Saved By", etc. }.

Both the DocumentProperties and DocumentProperty objects are selection and active-book independent.

The properties dialog can be brought up interactively using the Application.Dialogs(**xlDialogDocumentProperties**) constant.

To support customization of the property sheet, we define a new user-trappable event that is called whenever the document is about to be saved. The event is represented to the programmer by the **Workbook::OnPropertySheet** property, which is a string containing the name of the VBA Sub to run when the event is fired. The sub may bring up a custom property sheet form using a dialog sheet which is populated via the DocumentProperties objects. It is up to the callback to add, delete, or change any property values modified by the user. If the OnPropertySheet string is not defined, Excel reverts to the standard Summary Info dialog.

The OnPropertySheet property is probably not saved. It is up to the template/addin implementor to set it up using an Auto_Open trap.

The user's custom data form macro that gets called from the event can live anywhere in Excel: a separate addin, or on a macro sheet that is contained in the same book as the form itself. In Word

Note that simply trapping Auto_Close isn't good enough, since the properties dialog can be called from the File Menu or when saving a document for the first time. Adding a specific event avoids requiring every programmer to trap a menu command, a procedure which isn't very robust and is very difficult to get right.

Extra Frills Dept.: We could support Application::OnPropertySheet as a global means to setup a property sheet. Not required. We could also support Workbook:: and

CONFIDENTIAL

FL AG 0067436
CONFIDENTIAL

Application::OnSave as a notification that the document is about to be saved, and that any custom properties that need manual calculation should be updated. OnSave would also be useful to add-in vendors.

WordBasic Support

Here are the macro statements that have already been implemented. I think its a complete set. Are we missing anything?

GetDocumentProperty$(name$, fResolveLink)

Returns the value of the document property name$. If fResolveLink is true, the function will return the contents of the bookmark specified by the document property name$. fResolveLink is an optional parameter which defaults to false.

SetDocumentProperty(name$, value$, fLinked)

Sets document property name$ to value$, creates the property if it doesn't currently exist. If fLinked is true, value$ is assumed to be a bookmark. fLinked is an optional parameter which defaults to false

GetDocumentPropertyNameIndex$(i)

Returns the name of the i-th document property. The document property space consists of (the built-ins + custom defined). Numbering starts at 1 goes to n (n is built-ins + custom).

CountDocumentProperties

Returns the number of document properties. The built-ins + custom defined.

DeleteDocumentProperty(name$)

Deletes the document property name$. Fails if the property doesn't exist or if it is a built-in.

DocumentPropertyReadOnly(name$)

Returns true if the document property name$ is read-only, otherwise it returns false. Currently, only built-in properties are read-only.

XLM Support: Excel

It is unclear how we're going to be dealing with XLM in Excel '95, but it is likely we'll have to implement it internally. Here's the proposed XLM commands:

PROPERTY.ADD(name_text, type_num, value)

Adds or redefines a custom document property on the active workbook. Available in interactive form.

name_text: Required. Name of the property.

type_num: Optional. 1 for linked properties, 2 for static-value properties. Assumed 1 if omitted.

value: Required. The value of the property as text or a reference.

Redefining an existing property will change its value. Attempts to redefine a built-in property name (such as "Title") will fail.

HIGHLY CONFIDENTIAL

PROPERTY.DELETE (index)

Deletes a property on the active workbook. Note: if it is easier to implement deletion as PROPERTY.ADD(name) we should use it.

index: Required. The index of the property to delete, corresponding to the listbox location and the index of the property in the array returned by GET.WORKBOOK(*nn*). The text name of the property can also be used.

GET.WORKBOOK(*nn*)

Returns a horizontal array containing all custom properties of the active workbook, or #N/A if there are no custom properties.

GET.PROPERTY(name_text, info_type)

Returns information on custom properties of the active workbook.

name_text: Required. The name of the property to retrieve information for. If the named property does not exist, #N/A is returned.

info_type: Required. A number indicating what type of information to return:

1 The type of the property, 1 for linked, 2 for static.

2 The definition as text.

[3] Whether the property is built-in. (Whether this is publicly exposed depends on the OB implementation and whether built-in prop names appear in the property name array returned by GET.WORKBOOK(nn).

XLM To OB Mapping Information:

```
PROPERTY.ADD(NM, 1, VAL) : BindRef(VAL) =>
        {
        ActiveWorkbook.DocumentProperties.Add name := NM,
                type := xlLinkedRange, value := Range(VAL)
        }

// we need a separate rule for the named range case since I don't think the
// normal ref-translation handles names on the non-active sheet
// (i.e. Range("name") won't work).
PROPERTY.ADD(NM, 1, VAL) =>
        {
        ActiveWorkbook.DocumentProperties.Add name := NM,
                type : = xlLinkedRange,
                value := ActiveWorkbook.Names(VAL)
        }

// Should record textual property name for Index, not a number.
PROPERTY.DELETE(INDEX) =>
        {
        ActiveWorkbook.DocumentProperties(INDEX).Delete
        }

// On second glance, we should differentiate between adding and redefining for
// recording. Need a binder and separate recording rules.
```

CONFIDENTIAL

User Ed

Users may want to use this feature either for working with Notes, EMS or Elmer.

MSOfficeSummaryInformation need to be documented in the Office SDK.

Q & A

Which properties are promoted to the docfile, and then to the MAPI envelope?

All properties are promoted to the docfile and to MAPI.

Changes

9-Jan-94	First Spec	rgreene
1/20/94	New dialogs, scaled down proposal	rgreene
1/21/94	Added Excel issues, details on Prop storage and F/X, VBA and XLM implications	russellw
1/25/94	More details, close some issues	russellw
3/1/94	Filled in details.	keithg
3/29/94	Lots of changes. Added property set definitions, and property lists. Included new dialogs from DavidTu. Cleaned up overall. Refocused on OLE Property Exchange. Added external container issues.	mikemat
3/31/94	Took over the ownership of the spec. Added appendixes, combined 94 1/2 properties and new 95 properties in the same table. Assigned FMTIDs for MSOfficeSummaryInformation and UserDefinedProperties streams.	heikkik

HIGHLY CONFIDENTIAL

Appendix A – Standard Document Summary Information

OLE defines a standard common property set for storing summary information about documents. The Document Summary Information property set must be stored in an **IStream** instance off of the root **IStorage**; it is not valid to store the property set in the "CONTENTS" stream of a named **IStorage**.

All shared property sets are identified by a stream or storage name prepended with "\005" (or 0x05) to show it is a property set shareable among applications and the Document Summary Information property set is no exception. The name of the stream that contains the Document Summary Information property set is: "\005SummaryInformation"

The FMTID for the Document Summary Information property set is:
F29F85E0-4FF9-1068-AB91-08002B27B3D9

Use the DEFINE_GUID macro to define the FMTID for the property set:
DEFINE_GUID(FormatID_SummaryInformation, 0xF29F85E0, 0x4FF9, 0x1068, 0xAB, 0x91, 0x08, 0x00, 0x2B, 0x27, 0xB3, 0xD9);

On an Intel byte-ordered machine, the FMTID has the following representation:
E0 85 9F 4F 68 10 AB 91 08 00 2B 27 B3 D9

The following table shows the property names for the Document Summary Information property set, along with the respective property IDs and type indicators.

Property Name	Property ID	Property ID Code	Type
Title	PID_TITLE	0x00000002	VT_LPSTR
Subject	PID_SUBJECT	0x00000003	VT_LPSTR
Author	PID_AUTHOR	0x00000004	VT_LPSTR
Keywords	PID_KEYWORDS	0x00000005	VT_LPSTR
Comments	PID_COMMENTS	0x00000006	VT_LPSTR
Template	PID_TEMPLATE	0x00000007	VT_LPSTR
Last Saved By	PID_LASTAUTHOR	0x00000008	VT_LPSTR
Revision Number	PID_REVNUMBER	0x00000009	VT_LPSTR
Total Editing Time	PID_EDITTIME	0x0000000A	VT_FILETIME
Last Printed	PID_LASTPRINTED	0x0000000B	VT_FILETIME
Create Time/Date*	PID_CREATE_DTM	0x0000000C	VT_FILETIME
Last saved Time/Date*	PID_LASTSAVE_DTM	0x0000000D	VT_FILETIME
Number of Pages Number of Words Number of Characters	PID_PAGECOUNT PID_WORDCOUNT PID_CHARCOUNT	0x0000000E 0x0000000F 0x00000010	VT_I4 VT_I4 VT_I4
Thumbnail	PID_THUMBNAIL	0x00000011	VT_CF
Name of Creating Application	PID_APPNAME	0x00000012	VT_LPSTR

HIGHLY CONFIDENTIAL

[redacted]

FL AG 0067440
CONFIDENTIAL

Security	PID_SECURITY	0x00000013	VT_I4

* Some methods of file transfer (such as a download from a BBS) do not maintain the file system's version of this information correctly.

HIGHLY CONFIDENTIAL

HIGHLY CONFIDENTIAL

Appendix B – MS Office Summary Information

The Microsoft Office Summary Information properties are stored in a separate stream from the Standard Summary Information properties. The name of the stream for Microsoft Office Summary Information is:

"\005MSOfficeSummaryInformation"

The FMTID for the Microsoft Office Summary Information property set is:

D5CDD502-2E9C-101B-9397-08002B2CF9AE

Use the DEFINE_GUID macro to defint the FMTID for the property set: DEFINE_GUID(FormatID_MSOfficeSummaryInformation, 0xD5CDD502L, 0x2E9C, 0x101B, 0x93, 0x97, 0x08, 0x00, 0x2B, 0x2C, 0xF9, 0xAE);

The following table shows the added properties to the "MSOfficeSummaryInformation" stream for Office 95 applications.

Order of the properties in the stream

ISSUE: This is just a collected order of the properties. Need to go through and create an order that makes more sense if possible.

Property Name	Property ID	Property ID Code	Type	Notes
Project	PID_PROJECT	0x00000002	VT_LPSTR	NEW
Status	PID_STATUS	0x00000003	VT_LPSTR	NEW
Category	PID_CATEGORY	0x00000004	VT_LPSTR	NEW
No. Of ToDo's	PID_TODOCOUNT	0x00000005	VT_I4	NEW
ToDo's	PID_TODOS	0x00000006	tbd	NEW, Needed -- DavidTu/MikeCon?
Titles of Doc Parts[5]	PID_DOCPARTS	0x00000007	tbd	FROM AFX 94 1/2
Directory	PID_DIRECTORY	0x00000008	VT_LPSTR	FROM AFX 94 1/2
Document Class	PID_DOCCLASS	0x00000009	VT_LPSTR	FROM AFX 94 1/2 What is stored here? The human readable of classid or the version dependent

[5] This is SlideTitles for PowerPoint, WorksheetTitles in an Excel workbook, Subdocs in a Word master document or binder part names in binder.

				prog id?
File Name	PID_FILENAME	0x0000000A	VT_LPSTR	FROM AFX 94 1/2
Number of Bytes	PID_BYTECOUNT	0x0000000B	VT_I4	FROM AFX 94 1/2
Number of Lines	PID_LINECOUNT	0x0000000C	VT_I4	FROM AFX 94 1/2
Number of Paragraphs	PID_PARCOUNT	0x0000000D	VT_I4	FROM AFX 94 1/2
Launch Slide Show	PID_LAUNCHSLIDES	0x0000000E	VT_I4	FROM AFX 94 1/2
New Document	PID_ISNEW	0x0000000F		KeithG ????

HIGHLY CONFIDENTIAL

M CONFIDENTIAL

MS 0171063
CONFIDENTIAL

FL AG 0067443
CONFIDENTIAL

Document Wizards/Templates

Program Mgr John Misko

Summary Provide the user with a list of "business solution" Wizards to choose from when creating a new document. The greatest added value will be Wizards simliar to Money's Financial Wizards, but tightly integrated with Excel (for example, export the results and model to a sheet with on-sheet UI controls for continued what-if analysis). In addition to Money-like Financial Wizards, Wizards that create documents that will inherently benefit from an underlying calculation engine, like expense reports or work estimation forms, may be useful (we need to understand this for the Excel user and understand usage of same features in Works and Publisher). There is an Office 95 feature called "Visual Examples" which utilize CBT to show users how to use, for example, Excel features to create a document. These are document templates with "hot spots" on the template that describe to the user, with tips, how neighboring areas were created (see Visual Examples specification). We will make rational trade-offs between implementing as a Wizard or Visual Example (much less work, mabye the same benefit in the case of some templates, like an invoice).

HIGHLY CONFIDENTIAL

M CONFIDENTIAL

MS 0171064
CONFIDENTIAL

90

FL AG 0067444
CONFIDENTIAL

Justification

Document Wizards will improve Excel's usability by allowing the user to start work immediately, and by doing real work for the user. In addition, as Discoverability is an overall theme for '95 product releases, we intend to ensure that the Wizards make many of Excel's more advanced features available to the average user.

Description

The following is a list of proposed wizards/templates for Excel 95. In addition to this list, we can choose from the most successful wizards and document templates from Works, Publisher (Seven Business Forms) and Money and start with those, as a lot of work has been put into them already. Access to this functionality will be consistent across all Office 95 applications.

- Account Receivable
- Account Payable
- Amortization
- Break-even Analysis
- Budget
- Cash Flow
- Check Register
- Customer Invoice
- Customers List
- Employee Profile
- Equipment/Parts Inventory
- Expense Report
- Five Year Forecast
- Income Statement
- Job Estimate

Business Related

- Key Contacts
- Monthly Sales Goals and Performance
- Order Form
- Price List
- Project History
- Purchase Order
- Sales Goals and Performance-Yearly
- Salesperson Contact Sheet
- Service Invoice and Hourly
- Simple General Ledger
- Simple Stock Inventory
- Statement of Net Worth
- Straight Line Depreciation
- Weekly Time Sheet
- Work Order

Personal

- Book/CD/Tape/Video Library
- Collectibles Inventory
- Credit Card Log
- Household Inventory
- Membership Roster
- Monthly Calendar
- Personal Budget
- Personal Directory
- Personal Net Worth
- Shopping List
- Vacation Check List
- Personal Net Worth

HIGHLY CONFIDENTIAL

Platform	All.
International	The process of selecting the document templates to ship must take in consideration international usage (language, visuals, applicability).
Visual Basic	TBD.
User Ed	TBD.
Q & A	TBD.

Changes

4-Apr-94	First Spec	johnmis

HIGHLY CONFIDENTIAL

M CONFIDENTIAL

MS 0171066
CONFIDENTIAL

FL AG 0067446
CONFIDENTIAL

IntelliSearch 95

IntelliSearch is a feature which allows customers to enter help requests in a more natural <insert your native language here>-like fashion. Its main purpose is to map <insert your native language here> queries to the long list of help topics which the Microsoft Office will contain. Once the proper help topic(s) is identified, Intellisearch is responsible for dispatching to the proper content.

The input mechanism itself is really no more than an edit box which invites the customer to enter her help requests. This method of interaction should achieve four things: allow the customer a higher, more intelligent level of communication with Office (i.e. allows more valuable information to be exchanged between the customer and Office with less amounts of input work required); cuts down search time for help (in some cases the time savings would be infinite-fold since, previously, the customer may not have been able to find the required help at all); actively teaches the customer about the "Office world" by mapping their requests to Office specific features; and markedly improves discoverability and accessibility by intelligently bringing to the customer any appropriate, useful, and yet likely-to-be-underutilized functions and features.

This feature's design specifically aims at being a good cross-application feature for the MS Office package, and does so quite naturally as the problems which IntelliSearch tackles can be found in all MS Office products, and can be resolved in a similar fashion.

HIGHLY CONFIDENTIAL

Justification

1. As Office's functionality continues to grow, we require a better method for the customer to quickly and successfully search through and find help on desired topics. We can cut down search time dramatically by interpreting the customer's requests at a higher level and become more direct about launching them to the appropriate help. As opposed to the present help system, where the customer is required to navigate through several items to find help, we do all the navigating and searching for them and "surface up" all related help.

2. It is well known that Office customers utilize a relatively small fraction of the product. By building a better search/help mechanism we can have intermediate control (weighted probabilities) in the search process which allows us to intelligently "bring to the surface" those often underutilized, but highly useful related functions which so many of the customers are unaware of. This means that we do a better job of helping the customer complete their tasks. Intelligently bringing help topics to the surface really means two things: In one case we need to describe better how a particular function fits specifically into the customer's context, and in the other case, we need to associate and intelligently relate one function to other useful functions for the customer's task.

This is as opposed to the help that exists now, which, via keyword choice, generally drops you off in one "generic" help section. The problem with this is that the customer potentially loses other useful information and misses out on seeing how one Office function may be usefully tied into the "whole picture". Through intelligent mapping, it is possible to bring the customer a selection of various and related help topics which the customer may not have known or thought about.

3. The more a customer becomes aware of the previously unknown functions and capabilities of MS application software, the more they desire to stay with our product and recommend it.

4. Incorporating this sort of technology into Office demonstrates Office's commitment to using the latest and greatest technological advances to help the customer to get their work done faster and more intelligently.

[More customer research that lead to this feature is available; contact samh.]

HIGHLY CONFIDENTIAL

MS 0171068
CONFIDENTIAL

FL AG 0067448
CONFIDENTIAL

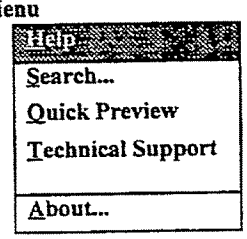
Description

Intellisearch (name TBD) is a central access point for online user assistance. It consists of both a "pseudo-natural-language" search and an online index (collectively referred to as "Intellisearch.") Customers use Intellisearch to specify what they need help with and Intellisearch dispatches the customer to the appropriate assistance.

Access to Intellisearch

Customers can get to Intellisearch via menu, toolbar, and keyboard.

Menu



launches Intellisearch dialog, defined below

launches the Quick Preview lesson

PSS contact information (troubleshooting information moves to Intellisearch)

Toolbar

Two buttons on the standard toolbar of each Office application:

button launches Intellisearch (button image tbd)

button loads the mouse cursor to context-sensitive pointer; next click summons QuickTips

Keyboard

F1 key (or COMMAND+/ or HELP on the Macintosh) launches Intellisearch (see "Context-sensitive Access to Intellisearch" below)

Shift + F1 key (or COMMAND+SHIFT+? or SHIFT+HELP on the Macintosh) loads the mouse cursor to context-sensitive pointer; next click summons QuickTips

Context-sensitive Access to Intellisearch

Accessing Intellisearch via F1 brings up the feature in context-sensitive mode. A context-ID is passed to Intellisearch to indicate the appropriate topics to display. If there is only one identified topic, the customer is dispatched immediately to that topic, and the Intellisearch dialog is never drawn.[6] If there is more than one topic attached to the

[6] If usability testing shows this to be ineffective, we may bring up the Intellisearch dialog in all cases, even if there is only one topic.

HIGHLY CONFIDENTIAL

M CONFIDENTIAL

MS 0171069
CONFIDENTIAL

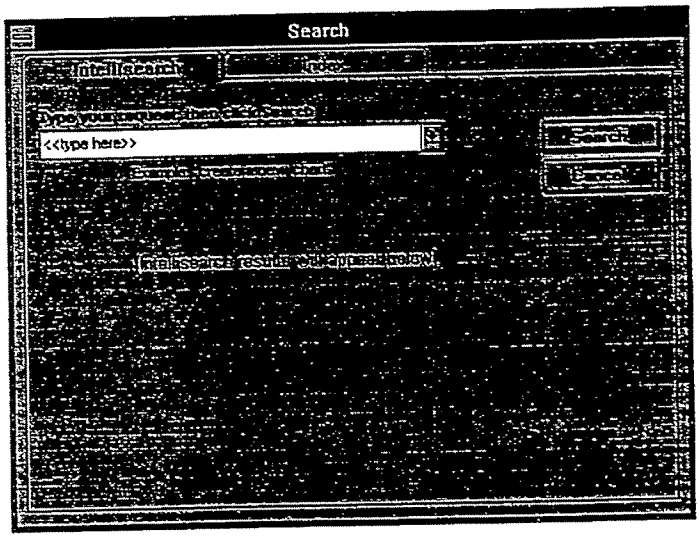
FL AG 0067449
CONFIDENTIAL

context-ID, the Intellisearch box contains the related text string for that context-ID[7]; this text is highlighted so that it disappears if the customer begins typing her own request. In context-sensitive invocations, the search results appear automatically in the lower half of the dialog (see below for description of search results section of the dialog.)

The exception to the above is when F1 is pressed when a dialog is up. In this case, a standard QuickTip is displayed that describes how to get the QuickTips in that dialog box.

The Intellisearch Tab

Intellisearch shares a tab dialog with the Index.[8] (The name for the dialog, "Search" is TBD.) The Index tab is included as a popular, proven, and more-traditional access method. It is also essential since Intellisearch is not available for all localized languages (see below.) The dialog is sticky, so the last tab chosen is current on the next invocation. In addition, the Intellisearch dialog contents are sticky (intra-session only), so the customer request-string and the results from the previous request are still in the dialog when it comes up.[9] The previous request string is selected so it disappears if the customer types a new request.



The Intellisearch tab before a request is entered

Entering a Request

The dropdown combo box contains past Intellisearch queries. The number of previous queries is an INI file option (and possibly exposed on an tab in Tools Options.) The

[7] Creating a text string for each context-ID is desirable from the customer's perspective, but it may be expensive for UE to create. If created, however, it would be useful for Testing's autotests....

[8] The Intellisearch tab looks huge because the Index tab behind it requires a large format for easier browsability. The visual design team is looking at the Intellisearch tab to improve its stark look. Ideas are welcome.

[9] This may be changed based on usability testing.

HIGHLY CONFIDENTIAL

FL AG 0067450
CONFIDENTIAL

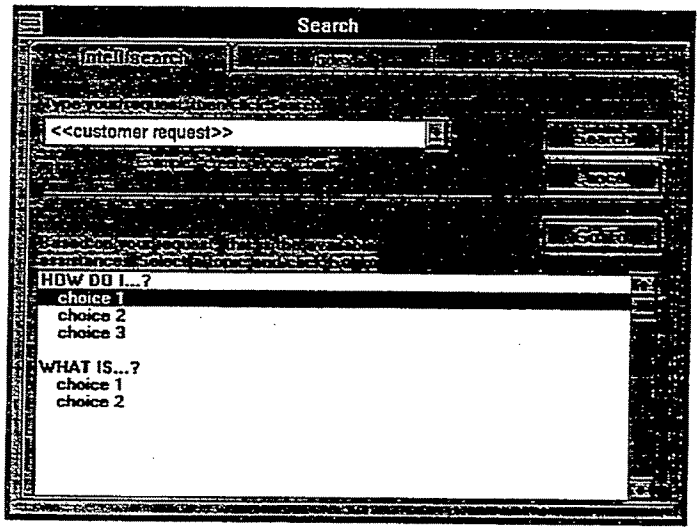
default is set at 10. The combo box contains a screened text entry "<<type here>>" that disappears when the customer types.

Intellisearch automatically spellchecks the entered string. The standard spellcheck dialog comes up on mis-spelled words. We need to make sure the Intellisearch synonyms are in the standard speller dictionary.

In certain cases, Intellisearch is invoked in a context-sensitive way, and a string is already entered in the combo box when the dialog comes up (see "Context-sensitive Access" section, above.)

Returning Search Results

After the "Search" button is clicked, Intellisearch returns the results in the lower half of the dialog tab.[10]



The Intellisearch tab after a customer request was entered

The returned "topics" are organized into two sections, "How Do I...?" and "What Is...?" Topics are identified by descriptive titles that fit in the "How Do I..." and "What Is..." construction.[11] This phrases the topics in the customer's terms and gives a breakdown between the two most-common assistance requests: procedural information and descriptions.[12] The returned topics will be limited to the top-seven "How Do I" topics

[10] It is important to re-inforce that the search results are coming from what was entered in the Intellisearch box above, so we need effects to make this clear to the customer. One idea is to have the search results "slide" down (like a garage door) from the horizontal line drawn below the example text. The visual design team is exploring other options.

[11] It is possible we discover topic titles are not always descriptive enough for customers to choose the appropriate one, so a linked Description box may be added alongside the topic list, but we will proceed under the assumption this Description box will not be needed.

[12] It is possible we will add one or two more sections, but we will try to fit everything into just these two sections unless an insurmountable problem arises.

HIGHLY CONFIDENTIAL

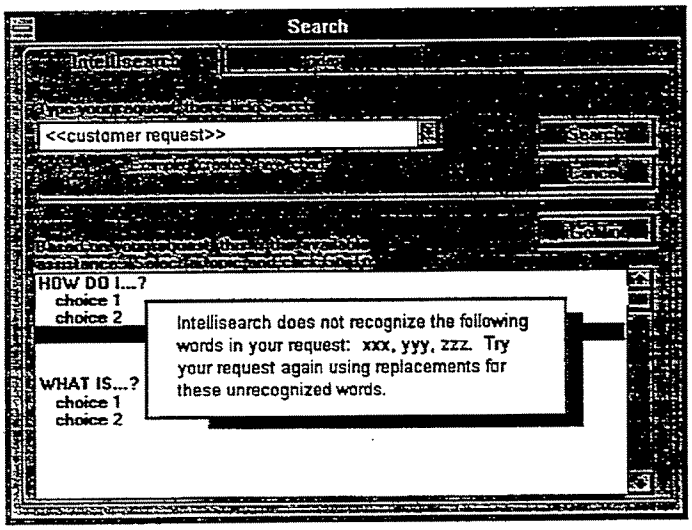
and the top-seven "What Is" topics (though it is unlikely there will ever be seven "What Is" topics. If this proves true, we might use the extra space for more "How Do I" topics.) This gives the customer a reasonable number of topics to browse and from which to choose.[13]

The returned topics are determined using the algorithm below (see "Technical Implementation of Intellisearch" and "Other Implementation Notes"), based on the probabilistic indexing we do at author-time, and they are presented in probability order. This means the top topic is the one we expect the customer is requesting.

Usability Testing

We'll want to test the Intellisearch returns UI/display for usability. Some existing research shows that customers often don't see the first item in a list because it is selected and hence reversed out. If customers don't see the first topic in Intellisearch (the one with the highest probability of being the one the customer wants), we'll change the UI to have none of the returned topics selected.

To handle cases in which the customer doesn't find the topic she wants, one topic is always returned at the bottom of the "How Do I" section: "Refine an unsuccessful search" (description TBD.) Clicking on this topic displays a popup that describes: (1) the mis-understood words and (2) a suggestion for how to fix the query. This topic moves up the probability list based on successive unsuccessful searches (implementation TBD.)



The popup for the "refine an unsuccessful search" topic

What happens when a topic is selected in Intellisearch?

[13] Once we get the topic list together, we may reconsider the seven-topic limit. For example, today there are about 40 topics on tables in Word. If the customer just types "tables", we may want to provide all the information we have. But, we may be able to write topics in a more-consolidated way to work with the seven-topic limit.

HIGHLY CONFIDENTIAL

When the customer selects one of the topics and clicks the "Go To" button (or double-clicks the topic itself), she is dispatched to the type of assistance related to the selected topic. The type of assistance we choose to provide is the "best way" of assisting, whether that means a long text description, a sentence of text, or a picture. The emphasis of Intellisearch is on *productivity,* not on instruction. This means we will keep the customer in the context of her work and in the product's main interface as much as possible. So, what are the possible assistance options? Clicking on an Intellisearch topic can dispatch the customer:

1. To a *QuickTip* (see separate QuickTips spec), "ghosting" the menu commands to get to the feature (similar to Lotus 1-2-3 Help in Excel.)
2. To a *wizard/feature.*
3. To a *cue card* (see separate Cue Card spec), listing the the necessary steps (a cue card is presented when a simple QuickTip is not sufficient.)
4. To a *visual example* (see separate Visual Example spec) with visual, conceptual information.
5. To a *WinHelp topic*, for conceptual or reference information (such as function descriptions.) [Note: Intellisearch must be able to dispatch to another HLP file as well, since the component nature of the Office means other groups will often provide us with HLP files.]
6. To a *pop-up definition.* Many "What Is" topics are best answered with a quick text definition. These are handled with a pop-up, similar to the pop-up definitions in the index for Word for Windows 6.0 and Excel for Windows 5.0.

One special "dispatch" Intellisearch can do is to return a number/code to the active application. This provides the flexibility for an application to handle special cases.

The type of dispatch is decided by UE at author-time, depending on which is the "best way" to provide assistance.

Following is a preliminary description of attributes for Intellisearch "topics." This is to help Development create the appropriate data structures.

Attribute	Attribute Description	Other Comments
Title (required)	Text description of the topic	This is how the topic is described in the lower half of the Intellisearch dialog box
ID (required)	Unique alpha-numeric identifier	For example "W123" is a Word-based topic and "E34" is an Excel-based topic. We can have a simple numeric ID instead, but tagging by product with an alpha character seems useful
Dispatch (required)	Indicates action associated with topic: 1-Show QuickTip (requires argument identifying the QuickTip)	This requires us to also specify unique identifiers for each QuickTip, cue card, visual example, and WinHelp topic.

HIGHLY CONFIDENTIAL

	2-Show "cue card" (requires argument identifying the cue card) 3-Show "visual example" (requires argument identifying the visual example) 4-Show popup (see below) 5-Show WinHelp topic (requires topic ID) 6-Show selection step, followed by one of the first 5 dispatch types	Regarding dispatch type #6: it will be important to have this capability when we want to use a dispatch type #1. (Have to figure out what this step looks like and acts like; probably a QuickTip look, but don't know how to dimiss it yet.) Do we need to support anything after a selection step other than a dispatch type #1?
Keywords (required)	A 3xN array containing keywords, probabilities, and (optional) link type	The link type is (D)efinite, (I)ndefinite, (N)eutral, or (U)nknown
Type (required)	A number 1- "How do I" 2- "What is"	Indicates which category to list topic under in the lower half of the Intellisearch dialog
Popup (optional)	Text (with optional bitmap) description of the topic (approximately 256 characters)	Some "How Do I" and "What Is" topics can be explained by a popup in the Intellisearch dialog; for these, there will be no further dispatch
IndexEntry (required)	Array of N text descriptions of the index entries corresponding to the topic (approximately 100 characters per array element)	This is what shows up in the Index tab. There can be many index entries per topic.
IndexPopup (optional)	Array of other topics' ID numbers	Some index entries will be "parents" for other, more-specific topics. Clicking on them pops up a few "children" topics to choose from. For example, clicking on the index entry of "examples and

HIGHLY CONFIDENTIAL

		demos" would pop up a list of a few topics to choose from. These can happen from both the Main Index and the Visual Index.
IndexParent (optional)	ID of another topic	If present, this tells which index entry to indent under
Visual (optional)	Bitmap	Required for those topics to appear in the Visual Index. Should this be a metafile?

Wherever possible, it would be nice to not have a hard-coded maximum-size for strings.

Cross-app Searches

In general, customers will use Intellisearch with the implicit expectation that the current application is the context (i.e. launching Intellisearch from within PowerPoint will yield assistance about PowerPoint and not about Word.) But, there will be cases in which a customer will ask for something that belongs in another Office application or involves multiple applications working together. This will be handled in the following ways:

1. The most-common cross-app activities (probably just a couple/few) are documented in each respective application (e.g. mail merge using Excel would be a topic in Word *and* in Excel)
2. The word/probability table of the current application (e.g. Word) is searched first. If no hits are found, the search continues on the MOM word/probability table. The MOM table contains words/probabilities for the top-10 single-application activities for each application and the top-n multiple-application activities (the kinds of activities DavidTu's MOM feature is designed to handle.)

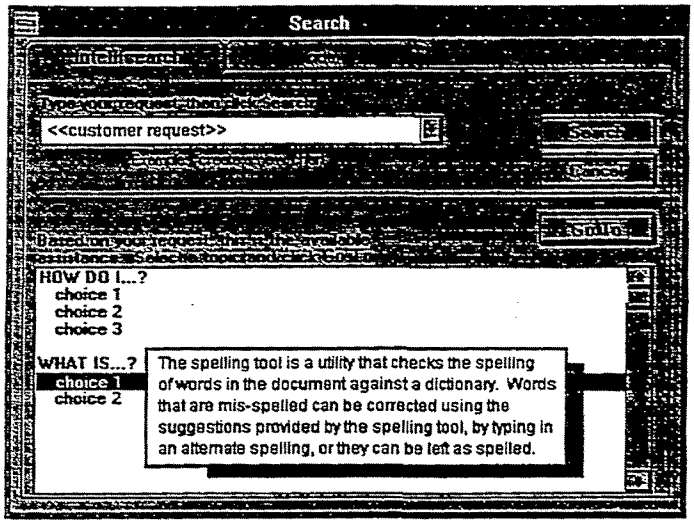
For example, if I'm in Word and my Intellisearch query is "do a mortgage amortization," Intellisearch would first go through the Word table. We'll probably have a topic for inserting an Excel grid in Word, so we may catch that topic if we have the words "mortgage" or "amortization." If we don't have those words, Intellisearch would go through the MOM table and would probably find an Excel-related topic via the words "mortgage" or "amortization." If not, Intellisearch goes to its failure mode (described above.) In Office 96, we'll take another look at a better solution to this problem, such as merging the tables into one.

Cross-app Dispatches

If there are any cases in which Intellisearch has to launch another application, we will ask the customer to confirm it before we do the dispatch. The context-switch is too large to do it without confirmation.

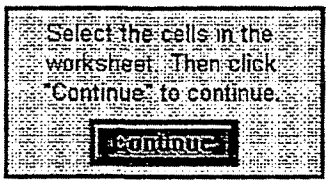
CONFIDENTIAL

The dispatches happen when the customer selects a topic and clicks on the Go To button or when the customer double-clicks on a topic. The pop-up definitions come up on a single-click of the topic. This is what a pop-up definition looks like:



A pop-up definition of the "choice 1" topic

Some dispatches, particularly those dispatches to QuickTips and wizards/features, require a "selection step" wherein the customer selects the data before choosing the command (e.g. Edit Copy.) Intellisearch provides a standard selection step before dispatching to the QuickTip/wizard/feature. The selection step takes the form of a QuickTip (see QuickTip spec) and is constructed in something like the form of "Select the ____ in the ____ . Then click "Continue" to continue." The variables depend on the application. For Excel, the text would read "Select the cells in the worksheet. Then click "Continue" to continue." (Exact text TBD by UE. The challenge is to be both descriptive and generic.)

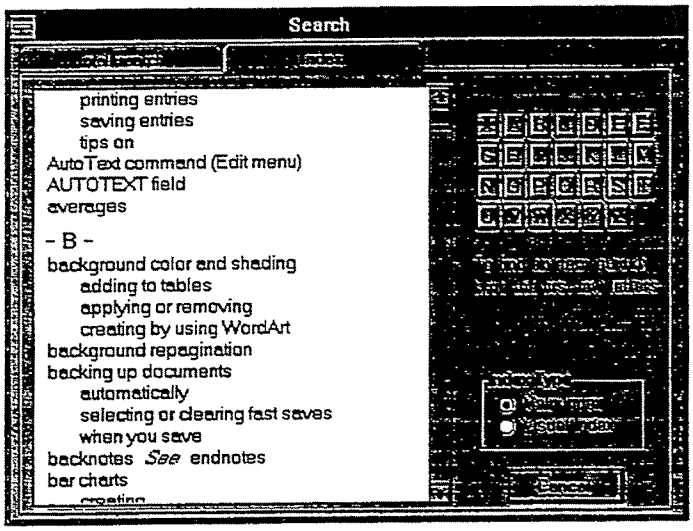


A guess at what the selection step might look like

HIGHLY CONFIDENTIAL

The Index

The Index tab looks like this:



The Index tab

The model for the Index is the Index that ships with Word for Windows 6.0 and Excel for Windows 5.0. Some notable features (the first two listed are part of the current Word/Excel indices; the others include new features to be added):

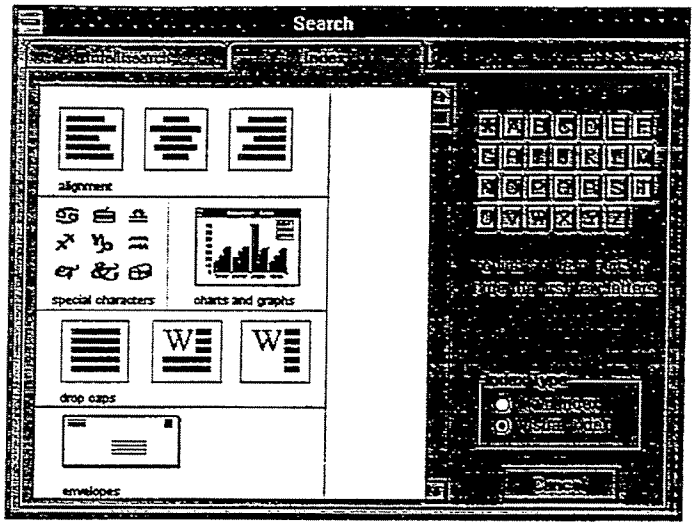
- Index is two levels only
- All items appear in the same color typeface
- Navigation controls: scroll bar, alphabetic buttons (the * button moves to the top of the index)
- Typing scrolls the index. The Capone type-ahead behavior is the model for the Index.[14] For example, typing "bar" for the Index above would scroll to the "bar charts" entry. Backspacing once (leaving "ba") would scroll the index back to the "background color and shading" entry. For the Index, type-ahead scrolls along only top-level items.

Visual Index
The visual index feature is currently being debated. It may or may not be included in the final design.

Clicking on the "Visual index" option button changes the view in the scrolling list box. The items in the visual index are a subset of the main index, higlighted here because they are easier to describe with a picture than in words. The same navigation behaviors apply as with the main index.

[14] We may need to add an edit field to re-inforce the type-ahead behavior. Usability testing will tell.

HIGHLY CONFIDENTIAL



The Index tab with the "Visual index" option selected

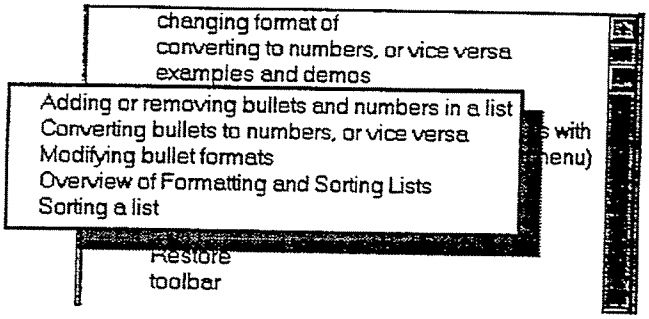
What happens when a topic is selected in the Index?

As with the Intellisearch tab, the items in the Index dispatch the customer to the assistance types listed above (i.e. QuickTips, cue cards, etc.) Some behaviors unique to the Index:

- When there are second-level items indented under a top-level item, only the second-level items correspond to individual assistance topics; clicking on the top-level item does nothing.
- The Index contains some internal jumps (e.g. "see 'Endnotes'" jumps the customer to the Endnotes item in the Index.)

HIGHLY CONFIDENTIAL

- Some second-level items displays a pop-up if the item corresponds to several topics; clicking an item in the pop-up takes the customer to that topic. (See example below for what an Index pop-up looks like.)



A pop-up list in the Index

Technical Implementation of Intellisearch:

First, we must know at design time what tasks and help topics we have to offer the customer. We will call this list of topics the concept list. Note that one concept might have "two faces". In normal usage, the concept may be a help topic, and in VBA, the concept could be the related methods, objects, and properties (e.g. "How to add a trendline" ==> Trendlines.Add). It is a matter of having a switch to determine which "form" of the concept gets returned. It is our first priority to create a word list which is arrived at by asking the following question: "Given that we know what concept the customer wants, what words would be used by the customer to suggest their desire to receive help on that concept?". The types of words which ***minimally*** should be used are: synonyms for any object, property, or method associated with the concept; words which Borland or Lotus would use for that concept; and words attained through study of novice to experienced customers in our user labs. Encoding this information is critical to the quality of the results obtained from this method.

Once we have lists of words for all of our concepts (many words will be non-unique, that is, a particular word might suggest multiple concepts), we need to assign probabilities to each (word,concept) pair. The question to ask here is: "Given that we know what concept the customer wants, what is the probability that the given word will be used by the customer to indicate their desire for help on that concept?". The notation we use for this probability is P(word | concept)

*It should be understood that assigning too many or too few words to a particular concept relative to the other concepts could be dangerous. **Research (and some simple mathematics) shows that a concept score is strengthened during a query more by the mere presence of several words pointing to it rather than the strength of the probabilities assigned to the (word,concept) pair.

One other note regarding our list of words. To keep the word list down to a reasonable size we will only use the base or lemma form of the word. When the customer inputs a series of words, those words are all broken down and sent to a stemmer. A stemmer is a morphological lexicon which groups words based upon their "base" form. That it is, words such as {run,ran,runs,running} are all grouped together since their base form is {run}. The stemmers we use are made available in-house and commercially (Houghton

M CONFIDENTIAL
MS 0171079
CONFIDENTIAL
FL AG 0067459
CONFIDENTIAL

Mifflin). The stemmer is designed to return the base form of the inputted word and this is what is used is our probability table. For example, {chart, charts, charted, and charting} all map to the single word "chart" which we will store in our word list. Not only does it keep our word list small, it also prevents the necessity to assign probabilities to every form of the word.

To obtain more information from the customer query, we exploit the notion that P(word | concept) depends upon whether or not the word applies to something which already exists vs. something which is desired. Software customers often wish to have assistance with transforming a current structure or objects into new structures. We have noticed that many free-text queries for assistance with specific types of transformations make use of words referring to (1) existing structures and (2) desired structures. We have found that it is useful in prioritizing the relevance of help on various transformations to discriminate words referring to existing concepts from words referring to desired states. In all Indo-European languages, and some other languages we'd like to support (English, Kanji, French, German, Spanish, Italian, Dutch, Swedish, Portuguese, Danish, Norwegian, and Finnish), particular "functional words" such as prepositions, possessives, demonstratives and articles suggest "definiteness" or existence of a particular object. I really mean "definiteness" in the grammatical sense. That is, words such as "the, my, or that" are definite whereas, words such as "a" are indefinite. Depending upon the concept, "definiteness" vs. the "indefiniteness" of a noun phrase will affected the penalization (if any) differently for the probability of each (word, concept) pair. When a customer desires help on a particular concept they tend to form their queries in a certain way. For example, when a customer wants to "make a new chart", they tend to use the indefinite form of chart (indefiniteness in this case is gleaned from the word "a"). On the other hand, if the customer wished to format their chart ("how do I format my chart") they will tend to use the definite form of chart (in this case, definiteness is gleaned from the word "my").

We can use this idea to improve upon the base method to improve the probabilities assigned to each (word,concept) pair. This makes IntelliSearch one step smarter since we can move beyond some basic problems inherent within a weighted keyword search (For example: given the query, "How do I format the text on my chart", IntelliSearch would not return the concept "Make a new chart", which would normally occur in a weighted keyword search). The attribute which needs to be assigned to each (word,concept) pair is what we will call the "word-concept link". We will allow the word-concept link to take on any one of four link types: definite; indefinite; neutral; and unknown. The question to ask when assigning the link is "What form of this word will be used if the customer does desire help on the corresponding concept?". I suspect that most links will be either neutral or indefinite because people tend to rely on words that refer to desired states when asking for assistance.

So, during the parse, we will group words according to the articular, possessive, demonstrative, and prepositional structure. For example, if A_i is any functional word, and B_i is any synonym from our word list, where $_i$ is the absolute position of the word in the sentence, (Note: When we scan the customer input, if the words in the query do not match our functional word list or our word list, they are thrown out as stop words.), then the grouping would occur as such:

For: $B_1\ B_2\ A_3\ B_4\ A_5\ A_6\ A_7\ B_8\ B_9\ A_{10}$

HIGHLY CONFIDENTIAL

$\Longrightarrow\ B_1 \mid B_2 \mid A_3\ B_4 \mid A_5\ A_6\ A_7\ B_8\ B_9 \mid A_{10}$

Because the rules we are using are distributive, we "distribute" A_i across B_i.

$\Longrightarrow B_1 \mid B_2 \mid A_3\, B_4 \mid A_5\, A_6\, A_7\, B_8 \mid A_5\, A_6\, A_7\, B_9 \mid A_{10}$

NOTE: There is an exception to the "breaking up" of single synonyms, and that is when multiple words are scored i.e. a certain sequence of synonyms is scored as one special word group. As the corpus gets larger and larger, multiple word scoring will become more and more important. The idea here is that if n adjacent words occur in the customer's query and they are recognized as a common phrase, they get scored higher. E.g. "line up", "chart background". The IDK has been designed to easily accept n word phrases. In effect, we are using a special case of word proximity to score the relationship between words, and hence their concepts, higher.

Another caveat, though in many cases you may be tempted to think that a particular sequence always points to a certain concept since it appears so unique, there will always be an exception to that rule, so make sure that at least the concepts for each word, individually, are scored, if only weakly. E.g. "I want to zero in on my data" might lead you to believe that "zero in" always means magnify the worksheet, though it could also be used in the sentence "How do I put a zero in my last decimal place?", which definitely points to another concept.

Many good places to use this technique are in cases where the customers specify a nonunique object along with a "domain resolver". I.e. "How do I format my chart text?", is perfect since the scoring for the concepts of formatting the worksheet can be weakened since "chart text" specifically resolves our higher level domain (see syntax matching in Q&A). The other places to use this are where a unique combination of words mean something totally different from each separate word meaning. I.e. "line up", "line chart", "cell box", "page number", etc. This part will require extra hard thought.

Let each word group be called a clause, and note that there is only one synonym per clause. For each article, demonstrative, possessive and preposition there is also a probability assignment, P(I), which is the probability that it indicates an indefinite structure. For any clause without a functional word, we give it a default value, say P(I)=0.6. Since their probabilities are independent, the probabilities of adjacent functional words, may be replaced with a single P(I), which is obtained by multiplying together all the separate P(I)'s. Effectively, all adjacent functional words are merged.

$\Longrightarrow B_1 \mid B_2 \mid A_3\, B_4 \mid A_{5\,6\,7}\, B_8 \mid A_{5\,6\,7}\, B_9 \mid A_{10}$,

where $P(A_{k\,k+1\,k+2\,\cdots\,k+n}(I)) = \&_{p=k}^{k+n} P(A_p(I))$

HIGHLY CONFIDENTIAL

For each B_i, at design time, we have determined whether or not the customer would use the definite form of B_i when requesting help on that item. The links will be identified as {definite(D), indefinite(I), neutral(N), and unknown(?)}, where the unknown(?) is the state that is assigned to a link when no assignment is made. Then, for each clause, we

apply the following formulae (derived from basic probabilistic axioms) for each concept which a word/synonym B_i points to:

Given the link {I, D, N or ?}:

L(I): P(clause|concept)=P(word|concept)*(P(I)+~P(I)*(leak term)^(1/n))

L(D):P(clause|concept)=P(word|concept)*(~P(I)+P(I)*(leak term)^(1/n))

L(N): P(clause|concept) = P(word|concept)

L(?): P(clause|concept) = ,*L(I) + -*L(D) + .*L(N) , where alpha, beta, and gamma are determined by the weighting of links in the word list table. So, alpha is just the number of indefinite links divided by the total number of links, and so on.

As mentioned above, there are several types of words that can play a role in the definite/indefinite links (i.e. the P(I)) The current types and the P(I) values that work well in prototyping are[15]:

Part of speech	Examples	P(I)
indefinite articles	a, an, some	0.7-0.8
definite articles	the	0.4-0.5
demonstratives	this, that	0.4
possessives	my, its	0.2-0.3
prepositions	under, on, by, above	0.2-0.3

HIGHLY CONFIDENTIAL

Meta-synonyms

When we create the Intellisearch keywords and synonyms, there will be special cases in which a broad, general notion has many virtually-synonymous forms. An example is the notion of "creation." For the purposes of indexing Office, many other words can imply the notion of creation: create, make, new, start, add, etc. Other "meta-synonyms" are "modification" and "deletion." There may be a handful of others. For these few meta-synonyms, we'll keep a list of their sub-synonyms. When one of the sub-synonyms are encountered in a customer question, the probability we use will be that of the meta-synonym. (Perhaps for a future version, we will want to fine-tune this arrangement by allowing the sub-synonyms to have "weightings" of how strongly the meta-synonym represents them. For example, maybe "make" is a pure synonym for "create," but start corresponds only at 60%.) This makes probability assessment easier (fewer words to assess) and eases localization. Note that sub-synonyms may serve two roles: (1) as a sub-synonym for a meta-synonym and (2) as a synonym of its own. For example, "add" is a sub-synonym for "create," but it should also point directly to the concept for "using the AutoSum button."

[15] There are many functional words we'll use for Intellisearch (right now we have about 100 in English); the table below just shows examples.

Calculating probabilities: the algorithms

Note that to simplify the assessment of synonyms, we will be using 13 probability buckets ranging from 0.1 to 0.9, obtained by multiplying 0.9 by (1/1.2)^b, where $0 \leq b \leq 12$, $b \in N$.

Furthermore, we will be computing penalties for (in)definite mismatch in terms of the leak probability, so as to assess the penalty in terms of numbers of mismatches on a list tolerated for every omission. Then, "n" is obtained by asking the question, "How many good word mismatches are equivalent to the omission of a good word?" Presently, setting n = 3 has worked quite well.

I have chosen a leak term of 1/100 since it smaller than the lowest (word,concept) pair probability by a factor of 10. This is the "punishment factor", and is used to punish words which do not all point to the same concept. In other words, if two words point to same concept, and only one of the words points to a second concept, the probability of (word | second concept) is divided by 100. This is due to the notion of strength in numbers (the number of words which point to the same concept).

These formulae are where the novelty of this system really stands out. More or less, what these formulae are saying is that if we expect the customer to ask for help on a particular concept using a word in a certain form, and that form is not used, then they do not desire that concept, and so it gets penalized.

Once this is calculated, we use idiot Bayes' formula.

In this case, at run time, we are given a set of clauses and wish to know what concept the customer most likely desires: P(concept | clause1...clausen). From Bayes' theorem and the idiot Bayes' assumption of conditional independence:

p(concept|clause1...clausen) = p(clause1|concept)* ... *p(clausen|concept)* p(concept) / p(clause1...clausen)

The denominator on the RHS is constant over all concepts, given the words that have been entered, and can be ignored if you just want rankings, which we do. If you assume that p(concept) is the same for all concepts, then you get p(concept|clause1...clausen) = K* p(clause1|concept)* ... *p(clausen|concept) ,where K is some constant.

The LHS is what you're computing at runtime, and the RHS is what you want to give at design time. So, when there is a set of words pointing to different topics, we simply multiply all P(word | concept) for like concepts and punish those probabilities which do not occur frequently. By doing this, we really say that if the majority of words in the set point towards one concept, it is most likely that that concept is the desired concept, and hence scored with the highest probability.

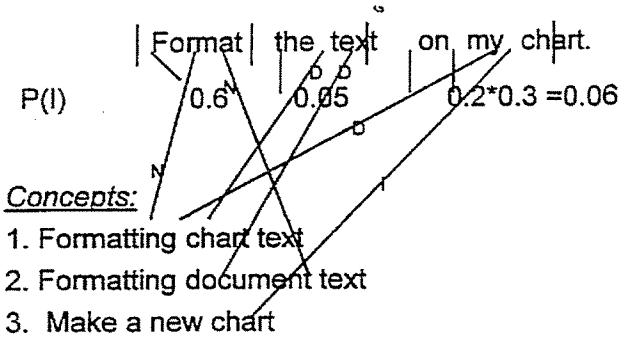
So, for each clause, we have one or more P (clause | concept) depending upon how many concepts the word/synonym in the clause points to. When we have the probability data we simply "resolve" them using the idiot Bayes' formula described above.

Example:

HIGHLY CONFIDENTIAL

MS 0171083
CONFIDENTIAL

FL AG 0067463
CONFIDENTIAL



Concepts:

1. Formatting chart text
2. Formatting document text
3. Make a new chart

Next we determine the score for each concept, for example:

Score for Concept(2):

=p("Format"|concept(2),neutral)*p("text"|concept(2),definite)*LEAK

=p("Format" | concept(2)) * [(0.05)*(1/100)^(1/3) + 0.95] * p("text" | concept(2)) * (1/100)

Now we just order the probabilities for each concept, and we have the concepts the customer most likely desires all ranked.

HIGHLY CONFIDENTIAL

Other Implementation Notes

1. The most-crucial part of developing this whole feature really lies in developing the best content for the thesaurus, and in determining the list of the most-likely-to-be-inputted questions and synonyms with their respective probabilities. We will require much help from User Education for gathering exhaustive synonym lists. In fact, UE will lead this effort, and a pilot project is underway. A modeling tool is available for gathering the synonyms and assigning them probabilities (i.e. the "Intellisearch Development Kit." See IDK.DOC on the spec server and the "User Education" section, below.)

2. The Intellisearch feature will be desired by other product groups at Microsoft. While we are not signing other groups up as clients, the code should be developed in a form such that the sources can be modified by other groups to meet their particular needs.

3. From an Office perspective, it is very important that we be able to look at help files from multiple applications, and we must also allow for ISVs and sophisticated customers to add their own topics and associated keywords/probabilities. A danger is that external entries can mess up our finely-crafted probabilities. There are some techniques for handling this. For each application, we have a set of probability buckets which have a particular number of word-concept pairs in them. Successful calibration can be done by "percentage-fitting". For example, a particular percentage of word-concept pairs are assigned to the highest bucket in Excel. In the case of integrating with Word, we simply grab that same percentage of word-concept pairs from the top buckets of Word and "drop" them into the highest bucket in Excel. In the case of third party vendors, we might choose to be a little "safer" and assign all the word-concept pairs from their application to the bottom, say 80%, of the Excel buckets. This type of calibration must be done every time new help is introduced into the customer's environment (either by another MS app, or from third party addins).

Customization
Need to design how customers and ISVs add Intellisearch topics. The VBA section gives some idea of this, but that information is out of date.

Hope you enjoyed this little trip into the guts of IntelliSearch!

Platform

HIGHLY CONFIDENTIAL

For any platform, the algorithm will be the same, though, we will require that the stemmer be supported for each platform. Houghton Mifflin has no problems with this requirement.

We may be doing the English stemmer in-house (the NLP group), though. It will be provided on a Win16 platform and it will be our responsibility to port it to Win32 and

Macintosh. The effort required to do this should be minimal, and the cost savings will be great (as opposed to letting HM provide us with the English stemmer, which they charge on a per unit basis).

International

Naturally, we would wish for this feature to be ported to different languages. This requires that we possess stemmers for each language we wish to support. Presently, Houghton-Mifflin is planning stemmers for English, French, German, Spanish, Italian, Dutch, and Swedish for summer 1994. Houghton-Mifflin is also considering stemmers for Portuguese, Danish, Norwegian, and Finnish. There are rumors of one for Kanji. We may be able to do the English stemmer in-house (via the NLP group), though, we will need to obtain licenses for the others from Houghton Mifflin. Those languages for which Intellisearch is not available will, instead of an Intellisearch tab, ship with a Search tab that works like the WinHelp Search in today's products.

Now, in many cases, gathering of the synonym lists will require more than just simple translation of the English synonym list and will probably result in a fair degree of work. It requires culturization on top of normal localization. It also means that whoever is in charge of their language will have to be very knowledgeable of their application and it's usage patterns to form significant word-concept links.

VBA

NOTES!
The VBA descriptions that follow are out of date with the spec and will be updated once the spec is more final. They are left here for later reference.

Application Class

HIGHLY CONFIDENTIAL

HelpFolders Method

This method returns a collection of all the top-level help folders in the active application.

Arguments: **index**

Optional argument. If omitted, all the top-level folder objects are returned, otherwise, just those folder instances specified by the index argument are returned. The index may be specified numerically or by using the name of the top-level help folder.

Returns: a collection of all the top-level help folder objects

Examples:

```
For Each HlpFold in Application.HelpFolders
    msgbox HlpFold.Caption
Next

Dim SpecialFold as Object
Set SpecialFold = Application.HelpFolders( Array(1, 3, "Help on Cells") )
```

RestoreHelp Method

This method restores the help to all the applications builtin help types only.

Arguments: none

Returns: nothing

Examples: Application.ResetHelp

HelpFolder Class

BuiltIn Property

HIGHLY CONFIDENTIAL

This property internally tracks which help folders are builtin (builtin means that they come from any MSapp!).

Access: read only

Data type: boolean

Apple event: 'fooo'

Examples:
```
If Application.HelpFolders(1).Builtin = True Then
  msgbox "This help folder is builtin!"
End If
```

Caption Property

This property corresponds to the friendly text description of the given help folder. Note that the caption of the help folder object is also the name of the help folder object.

Access: read/ write

Data type: string

Apple event: 'fooo'

Examples: Application.HelpFolders(1).Caption = "Making my document look great"

Add Method

HIGHLY CONFIDENTIAL

This method adds a new help folder. This may be a top-level help folder or a help subfolder of a higher-level help folder.

Arguments: caption, before

caption: Required argument. This is the text of the help folder (sets the initial value of the Caption property for the new help folder).

before: Optional argument. Specifies the help folder before which the new help folder will be added. Can be either a number or the caption of another existing help folder. Defaults to inserting the help folder at the bottom of the folder list.

Returns: a help folder object

Examples: Application.HelpFolders.Add caption:="foo", before:=3

Delete Method

This method deletes the specified help folders, and all contained subfolders and topics.

Arguments: none

Returns: nothing

Examples:

```
Application.HelpFolders(1).Delete

For Each HlpFold in Application.HelpFolders
    HlpFold.Delete
Next
```

HelpFolders Method

This method returns a collection of all the help subfolders under the specified help folder.

Arguments: index

Optional argument. If omitted, all the top-level folder objects are returned, otherwise, just those folder instances specified by the index argument are returned.

Returns: a collection of all the help subfolder objects

Examples:

```
For Each HlpFold in Application.HelpFolders(1).HelpFolders
    msgbox HlpFold.Caption
Next

Dim SpecialFold as Object
Set SpecialFold = _
Application.HelpFolders(1).HelpFolders( Array(1,3,5) )
```

Topics Method

HIGHLY CONFIDENTIAL

This method returns a collection of all the subtopics in the specified higher-level help folders.

Arguments: index

Optional argument. If omitted, all the topic objects are returned, otherwise, just those topics folder instances specified by the index argument are returned.

Returns: a collection of all the topic objects

Examples:

```
Dim counter%
counter% = 0
For Each Tpc in Application.HelpFolders(1).Topics
    counter% = counter% + 1
    msgbox Tpc.Caption
Next

Dim SpecialTopic as Object
Set SpecialTopic = _
Application.HelpFolders(1).Topics( Array (1, 3, 5) )
```

Topic Class

BuiltIn Property

This property internally tracks which topics are builtin (builtin means that they come from any MSappl).

Access: read only

Data type: boolean

Apple event: 'fooo'

Examples:

```
If Application.HelpFolders(1).Topics(1).BuiltIn = True Then
   msgbox "This topic is builtin!"
End If
```

Caption Property

HIGHLY CONFIDENTIAL

This property corresponds to the friendly text description of the given topic. Note that the caption of the topic object is also the name of the topic object.

Access:	read/ write
Data type:	string
Apple event:	**'fooo'**
Examples:	**Application.HelpFolders(1).Topics(1).Caption = "Making my document look great"**

Add Method

This method adds a new topic to the specified higher-level help folder in the active application.

Arguments:	**caption, before** caption: Required argument. This is the text of the topic (sets the initial value of the Caption property for the new topic). before: Optional argument. Specifies the topic before which the new topic will be added. Can be either a number or the caption of another existing topic. Defaults to inserting the topic at the bottom of the topic list.
Returns:	a topic object
Examples:	**Application.HelpFolders(1).Topics.Add caption:="bar", before:=3**
Notes:	**Change this so that words can point to multiple concepts. Should use array of structures.**

AddHow Method

HIGHLY CONFIDENTIAL

This method adds "How Do I" concepts and word links to specified topics in the active application.

Arguments: before, concept, synonyms, probabilities, linktype, linkname

before: Optional argument. Specifies the concept before which the new concept will be added. Can be either a number or the caption of another existing concept. Defaults to inserting the concept at the bottom of the concept list.

concept: Required argument. This is a string which appears as a topic under the "How Do I" heading.

synonyms: Optional argument. This is an array of strings which corresponds to the "How Do I ... ?" concept which appears when the topic is selected.

probabilities: Optional argument. This is an array of numbers [0.1, 0.9] which are the probabilities corresponding to the probability that a given synonym is used when a customer desires help on a particular "How Do I" concept.

linktype: Optional argument. One of xlAddin or xlHlp.

linkname: Optional argument. Path to help file or macro which should be invoked.

Returns: nothing

Examples:

```
Words(1) = "row"
Words(2) = "insert"

Probs(1) = 0.9
Probs(2) = 0.75

Application.HelpFolders(1).Topics(1).AddWhat _
before:=1, concept:="Inserting rows", _
synonyms:=Words, probabilities:=Probs, _
linktype:=xlAddin, linkname:="Insert_a_row"
```

AddWhat Method

This method adds "What Is" concepts and word links to specified topics in the active application.

HIGHLY CONFIDENTIAL

M CONFIDENTIAL

Arguments: **before, concept, synonyms, probabilities, linktype, linkname**

before: Optional argument. Specifies the concept before which the new concept will be added. Can be either a number or the caption of another existing concept. Defaults to inserting the concept at the bottom of the concept list.

concept: Required argument. This is a string which appears as a topic under the "What Is" heading.

synonyms: Optional argument. This is an array of strings which corresponds to the "What is ... ?" concept which appears when the topic is selected.

probabilities: Optional argument. This is an array of numbers [0.1, 0.9] which are the probabilities corresponding to the probability that a given synonym is used when a customer desires help on a particular "What Is" concept.

linktype: Optional argument. One of xlAddin or xlHlp.

linkname: Optional argument. Path to help file or macro which should be invoked.

Returns: nothing

Examples:

```
Words(1) = "row"
Words(2) = "rectangle"

Probs(1) = 0.9
Probs(2) = 0.3

Application.HelpFolders(1).Topics(1).AddWhat _
before:=1, concept:="What is a row", _
synonyms:=Words, probabilities:=Probs, _
linktype:=xlHlp, linkname:="c:\excel\row.hlp"
```

Delete Method

HIGHLY CONFIDENTIAL

This method deletes topics in the specified higher-level help folders in the active application.

Arguments: **none**

Returns: nothing

Examples:

```
Application.HelpFolders(1).Topics(1).Delete

For Each Tpc in Application.HelpFolders(1).Topics
    Tpc.Delete
Next
```

Help folders should be ranked by usage levels during adding and calibration!

Should make a TopicWizard to do this!!!

HIGHLY CONFIDENTIAL

User Education

There are numerous implications for User Education, and separate documents are available. The basic work item is creating a list of keywords and synonyms the customer may use to request each topic. Then, these keywords/synonyms need to be weighted by probability, as described above. (There is a tool to help with this indexing work: the IDK.) Contact Kristin Dukay for more information on Intellisearch indexing and the IDK.

Another basic work item is determining the "best way" to document each topic and writing/scripting that documentation.

Related to the context-sensitive access of Intellisearch, a short descriptive phrase must be created for each context-ID to be displayed in the Intellisearch box. This phrase must return the same topics as it does in the context-sensitive access if the customer types it in directly.

Questions And Answers

HIGHLY CONFIDENTIAL

Q: What happens when the search bottoms out or returns probabilities which don't surpass the threshold we set?

A: The best thing to do, at a particular threshold of uncertainty, is to have the system configured to ask the customer, "I am a bit confused about your query, can you describe your need in a different way?" (This will be handled in the special "Refine an unsuccessful search" topic that Intellisearch always returns. Then, we use the new query or *both* queries to generate a revised result. Also, this is the place to add the notion that we can add a little volume control to such querying of the customer for additional information or a rewording of his/her query.

Q: Why don't we use this at a systems level?

A: It would be a poor decision, competitively speaking, to give our competitors free access to our technology.

Q: How do we test this feature?

A: We are conducting a pilot test using Word for Windows 6.0. Kristin Dukay is leading the pilot; contact her for more information.

Q: What else can we do to enhance Intellisearch?

1. Query expansion and context sensitivity: we could do neat little things like make IT, CURRENT, ACTIVE, and SELECTED tied to the active object in the customers workbook.

2. Syntactic matching: we could do some brute force parsing of the customer's query and try to match it's structure to that of our object model. This is particularly useful for VBA. For example, if the query is "What is the method I use to add trendlines to my chart?", we could match this against "Charts.Trendlines.Add". If this was the strongest match, we would bump up the score for this concept. One might argue that this notion is already accounted for when assigning the probabilities at design time, but as the corpus grows, this type of gentle modulation is what helps us to further separate the probabilities of returned concepts, helping us to look smarter. One also never knows how well these concepts will be accounted for at design time.

Further, this type of parsing can help us to throw out "navigational" words. Often a customer will say "Do A method to B object", when A method doesn't apply to B object, but rather a subclassed object of B object. If we knew this, which we can in many cases, we could throw out word "B" and substitute it with the word for the subclassed object. This prevents our word weightings from being thrown off.

Changes

02-Dec-93	Authors of this spec are:	Jack Breese
		Erich Finkelstein
		David Heckerman
		Sam Hobson
		Eric Horvitz
		Greg Shaw
18-Aug-93	Erich Finkelstein	Created

HIGHLY CONFIDENTIAL

2-Nov-93	Erich Finkelstein	Description of Simple Bayes Model
28-Nov-93	Erich Finkelstein	Incorporation of definite and indefinite probabilities
13-Dec-93	Erich Finkelstein	Updated Excel Basic Issues
19-Feb-94	Sam Hobson	Added user-interface description, Index description, access method description, context-sensitivity description, and notes on implementation in Office
		Edited executive summary
		Streamlined description of algorithm
23-Feb-94	Sam Hobson	Updated access methods
		Added meta-synonym description, multi-application search & dispatch descriptions, and definite/indefinite P(I) values
27-Feb-94	Sam Hobson	Added ISSUES about spellchecking, usability testing
		Moved some descriptions into footnotes

HIGHLY CONFIDENTIAL

M CONFIDENTIAL

MS 0171097
CONFIDENTIAL

FL AG 0067477
CONFIDENTIAL

31-Mar-94	Sam Hobson	Refined context-sensitive (F1) access to Intellisearch Added spellchecking feature and removed spellchecking ISSUE Addes ISSUES for possible third tab and possibility of cutting Visual Index feature Added "failure mode" feature Refined Intellisearch dispatch types: removed "yellow brick road" and separated "wizard" dispatch from QuickTip dispatch Added Intellisearch topic attributes table Refined Index type-ahead behavior Added "selection step" feature to Intellisearch dispatches Added "sticky" feature for previous Intellisearch requests and results

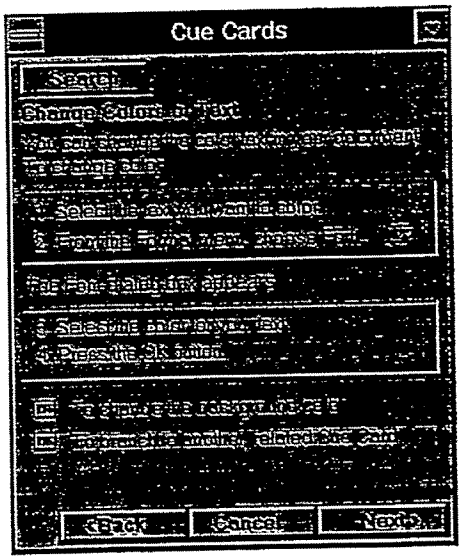
HIGHLY CONFIDENTIAL

Cue Cards

Program Mgr Stephen Hirsch

Summary Word95 will have Cue Cards as one form of assistance. They will be utilized in situations where the best help is step-by-step instructions that the user can follow to complete the task.

Cue Cards will be an integral part of the Office 95 user assistance strategy. They will be used when the user needs a step-by-step procedure to help her in the context of her own work. The content will come from our current Help Topic procedures, but will be used only when our Quick Tip approach is not adequate.



Sample Cue Card[16]

CONFIDENTIAL

16 The graphic design of this Cue Card has not yet been finalized. Eventually it will be changed to match all the other User Assistance.

Justification

Word needs to provide the user with improved assistance that aids the user in performing her *own* task. Cue Cards provide us with an opportunity to do just this, and with less printed documentation.

Cue Cards will be used in situations where we feel that step-by-step instructions are the best form of assistance.

Description

Cue Card Basics and Navigation

Cue Cards will cover most user assistance that are currently step-by-step procedures in Help Topics. Ghosting to QuickTips[17] will always be preferred to Cue Cards, and Cue Cards will only be implemented when the task is too complicated for Ghosting to QuickTips to handle. They will describe the steps to the user in a way such that she can easily follow along and complete her task.

Access

Cue Cards are accessed from the IntelliSearch or Index dialogs (and *possibly* Tip Wizard). See SamH's spec.

Appearance

All Office applications will have Cue Cards with consistent looks. The major characteristics are:

- Same color (gray or yellow, TBD) and size[18]
- Same feature set: occasional branching buttons, back, next and search.

Back, Next and Search Commands

The **Back** button will navigate back one Cue Card at a time (in the order the user progressed through them). At that point, it may function identically to the Search button. It is located in the upper left.

The **Next** button (when applicable) is located at the bottom right of the Cue Card and will bring up the next card for Cue Cards that demand more than one card.

The **Search** button (TBD) will launch the Office IntelliSearch dialog (See SamH Spec).

CONFIDENTIAL

[17] See Quick Tip spec.
[18] Need the advice of graphic designers – yellow or gray is just a guess.

Branching

We will attempt to limit branching as much as possible. There is some question as to the level of granularity IntelliSearch will map to. IntelliSearch plays the role of discovering the problem, so we will try to eliminate this task in the Cue Cards. There will be cases, however, where branching will still occur, but we will limit branching as much as possible. There will be no high level Cue Card Menu (except under "Cue Cards" in the Index).

In order to help limit branching, Cue Cards will only document the "best way" to do something. This way, we will not require branching to several different methods of accomplishing the same task.

Pop-up Definitions (maybe?)

When a Cue Card uses terminology that is confusing (i.e. Word jargon) it will be colored green and clicking on the text will offer a pop-up definition. Pop-ups can also be used to show a small graphic or offer a slightly better explanation when steps are potentially confusing to a large percentage of the users.

Platform All platforms.

International This will be a large part of all assistance, across all languages. Size of text in Russian and German could be a problem. It can be solved by using more cards (and more use of the "Next" button).

Visual Basic TBD

User Ed TBD

Q & A TBD

Changes

7-Dec-93	First Spec	SteveHir
17-Feb-94	Major Cutbacks and revisions for 95	SteveHir
30-Mar-94	Changes for final UA model	SteveHir

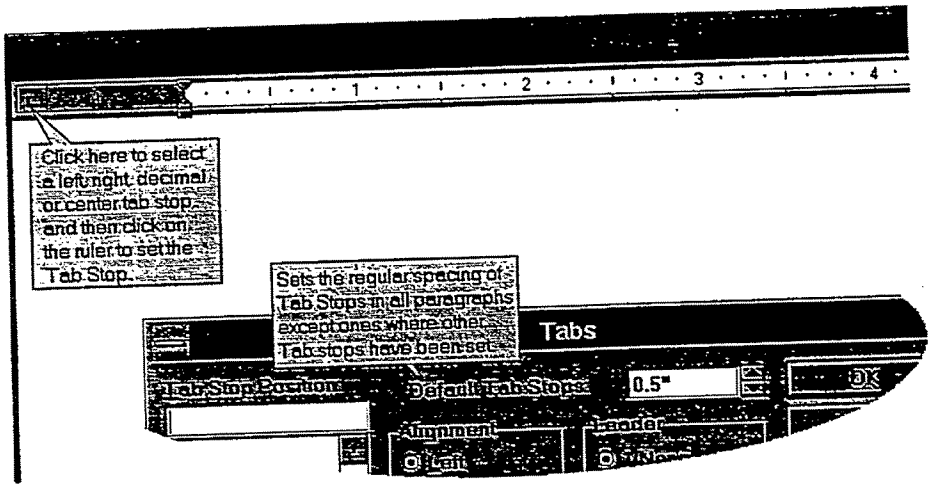
Quick Tips

Program Mgr Stephen Hirsch

Summary

Quick Tips deliver effective assistance in the context of the user's own workspace. They give brief descriptions of parts of our UI: buttons, controls, dialogs, etc. They provide much quicker access than standard Help dialogs and will decrease the total documentation.

Ghosting to QuickTips will be another way in which Office95 will leverage the work of these Quick Tips. For IntelliSearch[19] queries that need only to refer the user to a control with a simple one sentence explanation, the mouse pointer will "ghost" to the menu, then the menu item, and then the QuickTip will appear next to the relevant control. In this way, by simply selecting a topic, the user will arrive at the help he needs – quick, simple, and in context.



Examples of Quick Tips[20]

HIGHLY CONFIDENTIAL

[19] See IntelliSearch Spec

[20] The tails and visual design are not yet final.

Justification

Consistency/Leverage: Dialog Tips extend the Tool Tips feature currently in Word 6 and Excel 5.

Less documentation: implemented correctly, Dialog Tips can reduce documentation. Assistance in context can use less words *and* be more effective than assistance out-of-context.

Ghosting to QuickTips: This approach will leapfrog all current Help methods today. It takes the user to the desired place, so that the user will learn the product – and not be required to go through IntelliSearch next time. It also will be in the context of the users work. This is the best way to teach the user – as she is completing her own task.

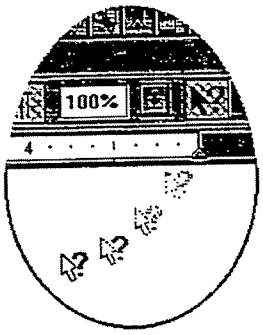
Description

Access[21]

There will be two methods of access for QuickTips. If the user needs help on a part of the standard interface (those things which are visible all the time), then she will need to load the mouse pointer and click on the control or area in question. If she needs help on a dialog, she will simply hit the QuickTip button that appears as she hovers over each control.

Loading the Mouse Pointer

Access to standard interface Quick Tips will be primarily through the Help button on the standard toolbar. This button will load the mouse pointer and activate tips when the left button of the loaded mouse is clicked (same as today's help button).



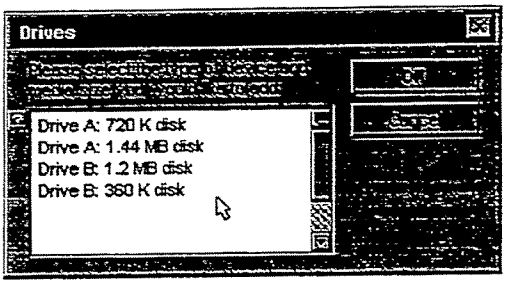
Loading the Mouse Pointer

HIGHLY CONFIDENTIAL

[21]Chicago and Office usability tests are underway, and we are anxiously awaiting the results of these and other tests to help us make a final decision on the best method of access.

Using the QuickTip Button in Dialogs

Within dialogs, as the user moves the mouse around, a small "?" button will appear next to the control that the mouse is over. The QuickTip will appear when the user clicks on this button.



Possible Chicago-style dialog

Keyboard users will access Quick Tips in tabbed dialogs by pressing Shift+Enter to load the cursor and then Enter on the appropriate control, or alternatively, Shift+Enter will bring up the Quick Tip for the currently active control.

Issue: How do keyboard users access non-dialog tips?
Do they need to? If they are keyboard users, they don't use these non-dialog controls.

Ghosting to QuickTips will be accessed via IntelliSearch and Index (see below).

Ghosting to QuickTips

Ghosting to QuickTips will, in many cases, be the easiest way to offer assistance. They will point the user to the appropriate control, without requiring the user to read any Help Topics or Cue Cards. Ghosting will be a great method for giving the user in-context help and allowing her to learn the product at the same time – the epitome of discoverability.

When IntelliSearch or Index dispatches the user to a QuickTip:

1. If the desired control is visible on the interface, it's Quick Tip will pop-up.

2. If the desired control is buried in a dialog, the mouse will automatically "ghost" to the menu, and then the menu is selected, the menu selection moves down the menu until the proper dialog item is selected, and then the QuickTip is displayed over the appropriate control.

Issue: What will appear to tell the user to select text?
See IntelliSearch for method of selecting text.

HIGHLY CONFIDENTIAL

[redacted]

FL AG 0067484
CONFIDENTIAL

Before any ghosting starts, it will first check to see that the toolbar button or menu item is in place (i.e. the user has not removed it.) If it is not present, a message will come up explaining that the item has been removed from the menu – or the dialog will simply appear without ghosting.

Integration with Tool Tips

Tool Tips will function as usual. However, Quick Tips for a toolbar button will have more text and more detail. Here are the cases where tips on toolbars will happen:

1. Hovering will bring up the Tool Tip, like today.
2. When the user loads the mouse pointer and clicks on a toolbar button, the longer Quick Tip will be given.
3. When the user chooses a topic from IntelliSearch, and we want to direct him to a toolbar button, the Quick Tip will be given.

Open Issue: Other methods of getting extended Toolbar tips.

Another possibility is that hovering gives a tip with a small ">" more button. This button gives the extended Quick Tip. Yet another idea is that if the user continues to hover the Tool Tip will expand into the Quick Tip. For now, this is just a record of some ideas, none are planned to be included yet.

Dismissing the Tip

Clicking anywhere outside a Quick Tip will cause it to disappear. Clicking on a Quick Tip will not make it disappear.

Appearance

Placement and design of tips are important aspects of Quick Tips. Below are the major problems to consider, and some suggestions for how they might be resolved (TBD):

HIGHLY CONFIDENTIAL

Text Boxes

Tips will appear in yellow callout boxes with black borders and black text. All boxes will indicate a direction so that they actually point out the button or control they refer to.

Location

The location of the tip boxes will be relative to the control. They will appear in a standard place relative to the dialog box. In the case of tips on moveable toolbars, they will appear relative to the button, but special considerations will need to be made so that tips don't go off the screen – we will do something similar to what was done for Tool Tips. We could also prefer a standard direction from the control, like "down and to the right."

Content of Tips

We will need to keep tips concise. There will be an upper limit to the number of characters that a tip can contain (TBD). For more complex features, there will be the possibility of a "More>" button[22] that will tunnel to more conceptual help (a Visual Example or Help Topic). In fact, all Visual Examples and Help Topics should be able to be accessed via the appropriate Quick Tip(s). Popups will be available from QuickTips, also.

Platform

All.

International

We should make sure that localizers can easily modify the strings in the tips. Balloons or boxes for the tips should autosize, and word wrap should be efficient.

Visual Basic

Tips for custom dialog boxes should be possible. Tips for custom toolbars should be possible. Perhaps 96?

User Ed

Needs to focus on being able to write clear and concise tips.

Q & A

TBD

Changes

7-Dec-93	First Spec	SamH
24-Jan-94	Rewrite for Word spec	SteveHir
17-Feb-94	Major trimming (and YBR) for Office95	SteveHir
25-Feb-94	Enhancements and Detail	SteveHir
30-Mar-94	Replacing YBR with Ghosting	SteveHir

HIGHLY CONFIDENTIAL

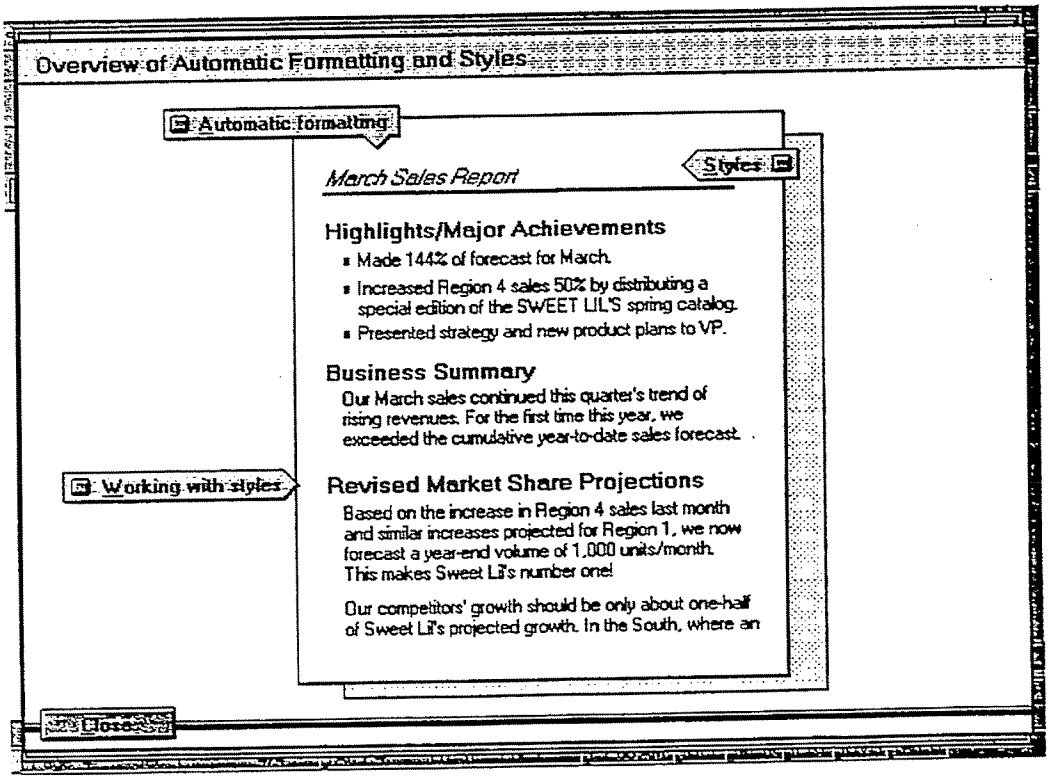
[22] Button image TBD.

Visual Examples

Program Mgr Stephen Hirsch

Summary Visual Examples will be a key part of our assistance strategy for all of Office95. Accessed from the IntelliSearch dialog, Visual Examples will be used when a conceptual overview is the best way to deliver assistance.

Visual Examples will be used for general topics that play off of our solutions. Memos, Letters, and Reports are examples of good Visual Example topics. In addition, overview topics related to using Word will be presented in Visual Examples: Viewing a Document, Customizing Word, and Document Management, for instance.



A Visual Example

HIGHLY CONFIDENTIAL

MS 0171107
CONFIDENTIAL

FL AG 0067487
CONFIDENTIAL

Justification

Visual examples are a key element of Office95's user-assistance strategy. They provide the best kind of assistance in certain cases, and they will be the only assistance available for help requiring a visual conceptual overview. This is in line with giving users the best available form of assistance based on the task, as well as limiting the amount of printed documentation.

Description

Visual Examples

Visual examples are large pop-up screens that show pictures (usually of finished documents) and have hot spots for pop-up explanations of document parts, and pointers to features. They are a terminal point to user assistance and do not offer jumps to any other forms of assistance.

Access

Access to Visual Examples will be via the IntelliSearch or Index dialogs (see IntelliSearch spec). The Index will have an "Examples" entry that gives the user a listing of all Visual Examples. It is possible that Tip Wizard tips or Quick Tips will tunnel to Visual Examples, which will be another method of access.

Content

Visual Examples will be used for conceptual overview information on: Memos, Letters, Reports, Document Management, Customizing Word, Document Viewing, and others (TBD).

Interface

The Interface for Visual Examples will be similar to Word 6 Examples. There should be no more than 5 or 6 hot spots per example. Pop-ups should clearly describe the feature or document element, and point the user to what feature she needs to do it for herself.

HIGHLY CONFIDENTIAL

Platform	Same across platforms
International	Lots of text and visuals to localize.
Visual Basic	TBD
User Ed	Authoring of content. Keeping focused on only using Visual Examples for conceptual overviews.
Q & A	TBD

Changes

26-Jan-94	Presentation	LisaMo
26-Jan-94	First Spec	SteveHir

HIGHLY CONFIDENTIAL

MS 0171109
CONFIDENTIAL

FL AG 0067489
CONFIDENTIAL

VBA 94

Program Mgr Russell Williams

Summary

Excel 95 will integrate and use VBA 94, aka the version of VBA that's used by VB4. VBA94 offers Excel programmer's the following improvements over VBA1.0:

- Data Access Objects are enabled.
- Speed improvements.
- 100% consistency with the dialect and runtime functions of VB4.
- UI enhancements cut from VBA93 that have been requested by customers (Indent, Outdent code, debugging enhancements).

HIGHLY CONFIDENTIAL

Justification

One of the top problems with VBA93 and Excel 5 is the lack of sophisticated programmatic data access. Excel offers the XLODBC addin based on ODBC, but this addin has reliability problems and is not the same as the data access mechanisms of Access and VB3/4. Use of VBA94 enables Excel95 to use the Data Access Objects (DAO), which offers hardcore programmatic data access based on Jet and a programmatic model identical to that of VB4.

VBA94 will be the "standard" version of VB that all Microsoft applications ship in 1994 and 1995. Excel95 needs to be in sync with VB4 and the other Office '95 applications to reduce customer confusion arising from differing dialects of VBA.

Lastly, VBA94 will fix bugs and improve performance problems noted with VBA93.

Description

The host integration model for VBA94 is not changing significantly, so replacing VBA93 with VBA94 mostly becomes a matter of rebuilding and implementing a few new menu items.

The spec describes the Excel-side changes required to integrate VBA94. To get details on language enhancements and VBA features not requiring any Excel changes see the VBA94 spec (CraigSy).

Edit and Continue

The most useful change in the UI is enabling of **Edit and Continue** in the Debug window while debugging. Excel 5 does not allow editing in the code pane of the Debug window while VBA code is on the stack. An alert is presented saying "You'll have to restart after this edit -- proceed anyway?", and pressing OK performs a Run / End followed by a Goto to the module containing the running code. Excel 95 will allow direct editing in the code pane based on rules defined by VBA94 (see the APBU spectool). At worst, editing some parts of the VBA code will require that a End be performed. For most edits execution can be continued.

Edit and continue support implies that Cut, Clear, and Paste items are enabled when the code pane of the debug window is active. Cut and Paste items are also added to the debug code pane shortcut menu.

Menu Items and Toolbar Buttons

VBA94 adds the following menu items to the Module menubar (exact layout TBD):

Run / Set Next Statement. Changes the "current" line marquee in the code pane of the debug window to be the selected line. When execution is continued by Continue or Stepping, execution continues from the newly designated line.

HIGHLY CONFIDENTIAL

Run / Show Next Statement. When in break (debug) mode, shows the line of code that will be executed next. Choosing this command may cause a different module to be opened and displayed in the code pane of the debug window.

Run / Step To Cursor. When in break mode, causes execution to continue from the current execution point and then break at the currently selected line. It is the functional equivalent of setting a breakpoint on the selection, choosing Continue, then removing the breakpoint. It is useful when debugging to skip loops or to quickly leave procedures that were accidentally stepped into.

Edit / Indent. Indents the selected lines of code by one tabstop.

Edit / Outdent. Moves the selected lines of code to the left by one tabstop.

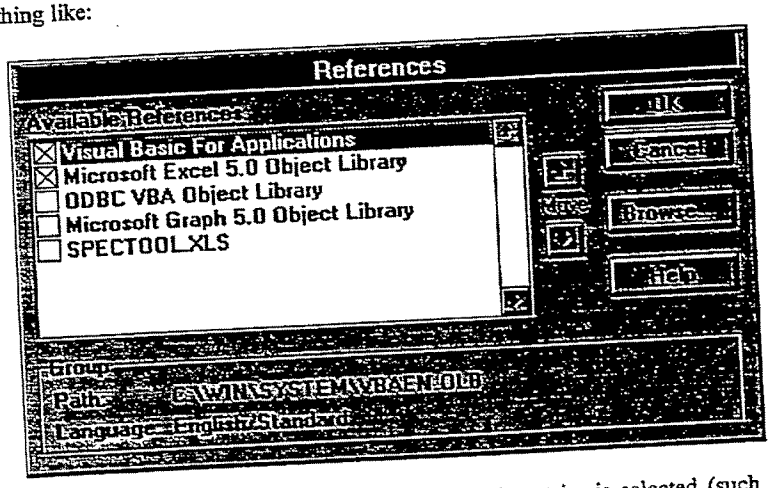
Show Next Statement appears on the immediate pane shortcut menu of the debug window.

Show Next Statement, Step To Cursor, and Set Next Statement appear on the code pane shortcut menu of the debug window.

New toolbar buttons are added to the Visual Basic toolbar for Indent and Outdent.

References Dialog

For consistency with VB4, the Tools / References dialog adds "Move Up" and "Move Down" buttons to change the ordering of typelib references. This allows the programmer to control the name binding order when using multiple projects. The new dialog looks something like:



The move buttons are grayed when one of the default entries is selected (such as the VBA or Excel typelibs). The move up button is also grayed when positioned immediately below the default entries, in this case when positioned on "ODBC VBA Object Library".

HIGHLY CONFIDENTIAL

Data Access Objects

VBA94 enables the Data Access Objects (DAO), the VB interface to Jet. Excel95 will ship DAO as the preferred mechanism for programmatic data access. See below for the user-ed implications.

Excel95 will continue to ship the XLODBC addin used by Excel 5 programmers for backwards compatability. The version of XLODBC shipped with Excel95 will be unmodified from the version shipped in the Excel5 maintenance release and will continue to use ODBC to get to external data sources. We will not attempt to map XLODBC calls onto Jet/DAO.

File Format

VBA94 uses a new file stream format. Since Excel95 will most likely be changing the file format as well this shouldn't cause a problem. VBA is doing the work to read and write VBA93 files and will supply new APIs to specify the file format when saving a project. These new APIs need to be hooked up in Excel so that the File Save As dialog specifies the correct VBA file format when saving Excel5 vs Excel95 files.

Platform

DAO and Jet don't exist on the Mac. This isn't an issue for Excel95 because we aren't doing a Mac release. It will become an issue for Excel96.

International

There should be no new International problems. DAO may cause some problems in that it has to recognize international date settings and sort orders, but these issues should be solved by DAO/VB4 before they affect Excel.

There may be a problem getting DAO localized into all the languages required by Excel. We need to get a list of DAO languages and determine discrepancies.

Visual Basic

No direct implications. Excel 95 will continue the policy of not allowing programmatic control over the module sheet, thus the VBA94 enhancements will require no addtional methods.

DAO will supply a set of standalone OLE objects that implement their own properties and methods. The only work that Excel needs to do make these objects available to VBA programmers is to register the DAO typelib and DLL.

HIGHLY CONFIDENTIAL

Setup Implications

Setup needs to register DAO's typelib and DLL. Jet-specific setup needs will be covered in a separate spec.

VBA94 DLLs have different names from the VBA93 DLLs. When setting up over an Excel 5.0 installation, Excel95 setup should offer to delete the VBA93 DLLs as an unused component.

User Ed

VBA94 will presumably come with a completely new set of "core" docs from APPU. We'll have to integrate these docs (remove VB4-isms such as references to forms in text and examples, VB4-specific menu commands, etc).

We'll need to integrate docs on DAO. Again, presumably these will be written by APPU (or somebody in DDT). The existing Excel95 topics on XLODBC need to be marked as obsolete. At least one new topic will be created to map XLODBC calls onto the DAO equivalent.

Q & A

This is great, but what does it cost?

Development estimates that integrating VBA94 will take about 2.5 man-weeks. The testing cost is higher, approximately 8 man-weeks assuming the VBA group completes file format backwards-compatability testing before handing off to Excel.

Since DAO uses Jet that means that Excel95 also has to ship Jet?

Yes, Excel 95 will ship Jet for DAO but we were already planning to ship it as a part of the ODBC drivers for ISAM database formats such as dBase, Fox, and Paradox. The only additional overhead of using DAO is shipping DAO2032.DLL.

VBA94 will use the same DLLs as VB4?

Yes, our goal is to ship exactly the same bits as get shipped with VB4. If there are post-VB4 bug fixes, though, we will use them.

Changes

5-Apr-94	First Spec	russellw

HIGHLY CONFIDENTIAL

M CONFIDENTIAL

MS 0171114 140
CONFIDENTIAL

FL AG 0067494
CONFIDENTIAL

Data Access Plumbing

Program Mgr Russell Williams

Summary

This spec describes the external data access technologies and pathways used in Excel 95.

Excel 95 will be a transitional release for data access. All built-in Excel data access features will continue to use ODBC, as they did in Excel 5.0. This includes Pivot Tables and the Get External Data command. MSQuery will continue to use ODBC as its data access mechanism.

ODBC's access to ISAM data such as Access (Jet Red), dBase, Fox, etc. will improve due to improvements in the underlying ODBC drivers. The ODBC group will replace the SIMBA-based ISAM drivers with drivers based on Jet. We will use these improved drivers, but the mechanism through which they are accessed is ODBC.

Excel 95 will improve programmatic data access by making the Data Access Objects (DAO) available to the VBA programmer. These Data Access Objects are the same objects that VB4 and Access2 makes available. Unlike the built-in Excel features, DAO goes directly to Jet, using ODBC only to access external (non-ISAM) data. XLM programmatic data access remains unchanged from Excel 5.0.

HIGHLY CONFIDENTIAL

M CONFIDENTIAL

MS 0171115
CONFIDENTIAL

141

FL AG 0067495
CONFIDENTIAL

Justification Explain why we're doing this feature here.

Description Excel 95 will be a transitional release for data access. Most of Excel's data access features will continue to rely on ODBC to get to external data sources. Excel 95 will get improved ODBC speed by use of Jet-based drivers instead of SIMBA-based drivers. Excel's programmatic data access from VBA will be vastly improved by the use of the Data Access Objects (DAO). A summary illustration is given in figure 1.

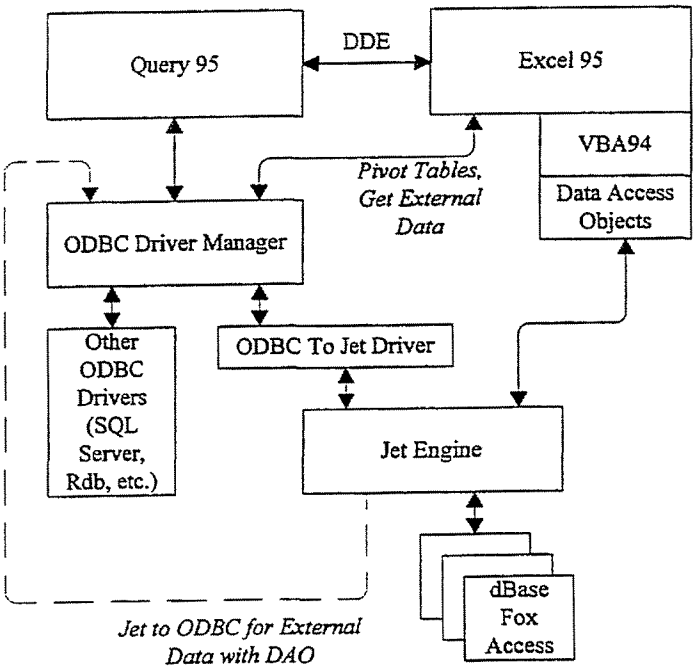


Figure 1: Overall Excel 95 Data Access

HIGHLY CONFIDENTIAL

Built-in Excel Data Access

The ODBC-based data access features of Excel 5.0 continue to use ODBC in Excel 95:

- PivotTables using external queries
- MS Query
- Data / Get External Data (even though this is an add-in, it relies heavily on MS Query and should use the same data access technology as MS Query)

Excel and Query 95 will not use DNA as its data access mechanism because DNA will not be available in our timeframe (our code complete is in late summer '94). Excel and Query will also not rewrite to use the Jet APIs instead of ODBC because of the impending arrival of DNA in the '96 timeframe. DNA should completely replace the Jet APIs as a functionally equivalent but more compatible and transparent interface, and it

doesn't make sense to rewrite our data access features in '95 if they are going to be replaced again in '96.

Lastly, there is the cross-platform issue to consider. Jet will not be available on the Mac in the Excel 95 timeframe whereas Mac ODBC exists today.

A simplified version of figure 1 is shown in figure 2. This version deletes the boxes not used by the built-in Excel features.

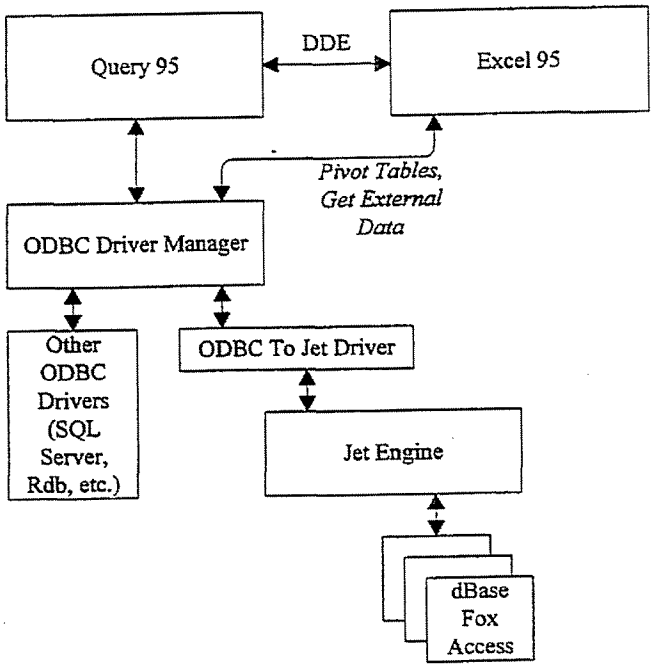


Figure 2: Excel 95 Built-In Data Access Features

HIGHLY CONFIDENTIAL

Jet-Based ODBC Drivers

Excel 95 will change the technology used by the ODBC drivers for ISAM data such as Fox, dBase, Access, Text, and Paradox. Excel 5.0 uses the SIMBA query engine to map SQL onto these file-based database formats. Excel 95 will use Jet as the query engine.

The newer ODBC drivers will be transparent to the user in that the interface used to get to them is still ODBC, and the query is still constructed in ODBC-standard SQL. The main advantage of the newer drivers is that Jet is faster than SIMBA. Secondary advantages are a reduction in hard drive space for the Office (only one database engine instead of two) and internal consistency with Access in how the query is run (so the same optimizations that work for Access will also work for Excel).

The ODBC group is producing the newer ODBC ISAM drivers in concert with the Jet group. The drivers (scheduled to ship as a part of ODBC Driver Pack 2.0) will be

available in late summer. Native 32-bit versions of the ODBC driver manager and Jet will be used.

Programmatic Control

The area of greatest change in Excel 95 data access is programmatic data access. Excel 95 will use the Data Access Objects (DAO) produced by DDT as the programmatic data access mechanism of choice. DAO appears to the programmer exactly like the data objects in VB4 and Access2.

DAO is enabled by VBA94, which Excel 95 will be using. There is no work involved for Excel other than shipping the DAO DLLs and writing the proper registration entries on the registration database. Excel's VBA documentation will need to include DAO documentation (presumably we can use the docs shipped with VB4).

The data access picture for DAO (omitting the built-in Excel features) is shown in figure 3.

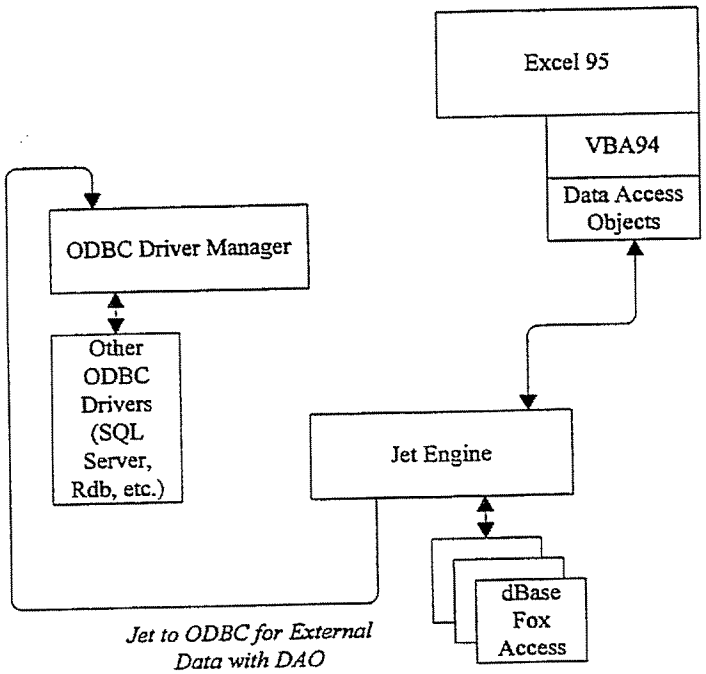


Figure 3: Programmatic Data Access

Note that programmatic data access and Excel's built-in data access takes entirely different paths to get to data. DAO will talk directly to the Jet engine. Jet will in turn use ODBC to get to external data when required (such as SQL Server). Excel's built-in features always go through ODBC and transparently use Jet when required. The result is a disconnect that will be difficult to explain to customers.

HIGHLY CONFIDENTIAL

Issue

We need to pinpoint exactly what the differences between DAO and ODBC will be. Known differences: Jet doesn't use DSNs and will be able to use stored queries in Access MDBs to generate result sets. Jet can do updatable cross-table joins. ODBC always uses DSNs, can't use stored queries, and can't do updatable cross-table joins. The security model is also very different.

For backwards compatability Excel 95 will continue to support the XLODBC addin functions shipped with Excel 5.0. The recommended method in Excel 95 for all new applications to get to data will be DAO.

Platform

Jet and DAO will not be available on the Mac in the '95 timeframe. Assuming there is a Mac release of Excel 95 this won't cause a problem for built-in Excel features, since we don't ship any ODBC ISAM drivers on the Mac other than Fox. The Fox driver does not use Jet. Since DAO won't be available we'll be leaving Mac programmatic data access customers out in the cold.

This issues needs to be tracked for Excel 96, where we are doing a Mac release and where a compatible cross-platform solution becomes very important.

International

Track the availability of localized DAO. If DDT is not going to provide localized versions we will either need to localize it ourselves or ship partially translated docs that describe the DAO objects in the localized language but have VBA syntax examples in English.

User Ed

Explaining the disconnect between DAO and ODBC will be a problem.

We will need to incorporate new VBA reference material to cover DAO, and modify the existing XLODBC topics to mark them as obsolete.

Q & A

Why aren't we using DNA?

DNA won't be available in the '95 development timeframe.

Why aren't we writing directly to Jet?

We don't want to rewrite to use Jet because we will update to use DNA in the '96 release. Rewriting twice is a waste of effort.

Will Query see any improvements: speed, more integration with the worksheet?

We are not planning any improvements to the Query UI for '95, again because we plan to hand off Query in '96 to Access and do much more integration work to tighly bind Query onto the worksheet and PivotTables. We are planning to make it easier

HIGHLY CONFIDENTIAL

FL AG 0067499
CONFIDENTIAL

to access and run canned queries in '95 with the AutoQuery feature, described in a separate spec.

Changes

5-Apr-94	First Spec	russellw

HIGHLY CONFIDENTIAL

M CONFIDENTIAL

MS 0171120
CONFIDENTIAL

FL AG 0067500
CONFIDENTIAL