Web-like Shell : Architecture

Internet Explorer Integration, In-place Navigation and Page-View

November 8, 1995
Satoshi Nakajima (satona), Windows UI Development PSD

Table of Contents

Revision History:

1.00 September 29 -- Written

1.1 October 27 – (1) Added "In-place navigation of Office documents on file system" as one of primary goal. (2) Change in History mechanism. (3) Updated the development status. (4) Updated the status on peer-to-peer HTTP server.

1.2 November 8 – (1) Updated pictures reflecting the latest mechanism. (2) Updated the mechanism to generate HTML pages for pageview.

Overview

As ChristoB describes in his "web-like shell" UI document, we will improve the Shell Explorer UI by making it very easy to "navigate" – with the navigation toolbar and the single-click page-view. We will also integrate the Shell Explorer and the Internet Explorer, so that the user can navigate documents on local volumes, LAN and WWW as seamless as possible. This document describes the overall architecture of this new shell.

Major goals in this release:

- *Integrate* the Shell Explorer and Internet Explorer
- Enable the *in-place navigation* of Office 96 documents on WWW and on file system
- Make the shell *behave and look like a web-browser* (navigation toolbar and single click page view)

Notes [Oct.27]: After talking to the Office team (RWolf and SriniK), it became clear for me that in-place navigation of Office (and Office-compatible) documents on file systems is as important as the Internet senario. The users should be able to build a web of documents no matter which protocol (HTTP or file) they choose and no matter which format (HTML, DOC or something else) they choose.

Wishes:

- Enable the in-place navigation of Office 95 documents as leaf nodes as well
- Make the shell folder view of shell name space extensible (for third party shell)
- Make the page view extensible (for third party shell)
- Enable the embedding of shell folder views in OLE containers (esp., mail messages)
- Make the shell browsing fully remotable (over LAN, Internet, modem and IR)
- Put a Gibraltor-compatible peer-to-peer (personal) HTTP server on every Win96 machine

Restrictions/Constraints:

- We need to minimize the changes to the existing code path
- HTML view control won't be ready soon
- Hyper-link aware Office 96 apps won't be ready soon
- All the design work should be done as soon as possible

Strategy:

- Use existing interface layers (extension mechanisms) as much as possible
- Minimize the change to existing components (explorer.exe, shell32.dll, comctl32.dll)

- Modularize components for parallel development and component testing

Existing Interface Layers

To achieve our modularized development, we'll utilize the existing interfaces as much as possible. It will enable us to (1) minimize changes to existing code path, (2) test individual components independently and (3) minimize the dependencies between groups (Shell, O'Hare, Office96). We'll use following interfaces:

Windows 95 Shell Namespace Extension:
Although we haven't clearly defined how we presents documents on WWW to the end user on the Explorer left pane (i.e., the hierarchy), we know that they don't belong to any of existing folders (shell's namespace). It is quite natural to use the Namespace Extension mechanism (see picture below) to plug the URL namespace into the explorer's name space.

Office 95: DocObject interface:
Office 95 introduced the DocObject interface for Binder and Word-As-Mail-Editor. In-place navigation of Office documents (and HTML documents) can be achieved with this mechanism with a minor extension (hype-link interface, which Office 96 will support) to it.

OCX/OLE embedding:
OCX is already "the" interface to integrate arbitrary UI components into form-like container. It is a very natural choice and strategically makes sense for us to support OCX extension in its HTML viewer. To achieve our UI requirement (having a full functional shell view in a page view), we need to create our own window in a HTML page, and OCX gives us enough flexibility.

BGI:
BGI (Binary Gateway Interface) is an extension mechanism, which the Gibraltar group defined as the interface between the Gibraltar HTTP server and extension DLLs – those DLLs will generate (mostly HTML) documents on the fly. If we decide to "generate" page-view HTML documents on the fly rather than using fixed HTML files, it is clearly wise to use this mechanism.

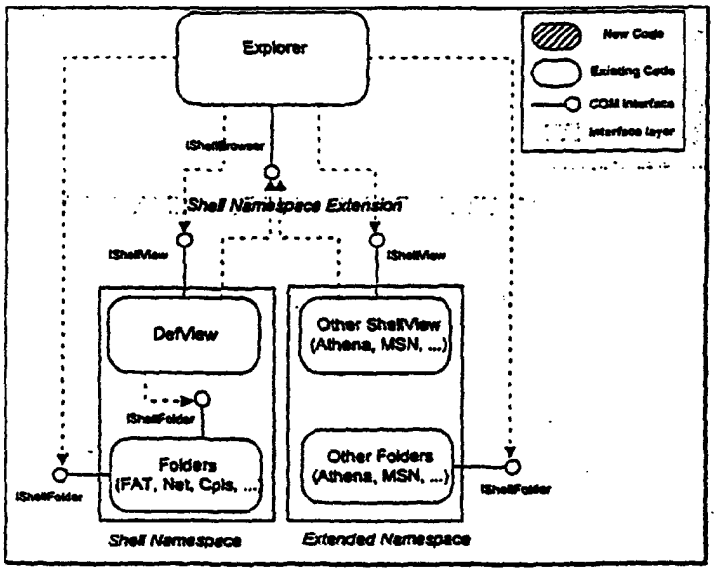


Fig.1 Windows 95 Shell Namespace Extension Mechanism

MS9 000033
CONFIDENTIAL

MS98 0116190
CONFIDENTIAL

MS-PCA 1085016

MS-CCPMDL 000000226888
CONFIDENTIAL

Shell Explorer - Internet Explorer Integration

Fig.2 shows you how we will *integrate* the Shell Explorer and the Internet Explorer. At the same time, we will enable the *in-place navigation* of Office documents and add web-browser like navigation UI to the Shell Explorer. There are five pieces of either new of modified components to achieve all of these goals.

DocView object (owned by SatoNa -- Shell):
To "host" (i.e., in-place navigate) Office documents (both 95 and 96), the shell needs to become a DocObject container. It is theoretically possible to change the explorer code to directly support the DocObject interface, it requires significant changes to the existing code and the risk of breaking existing behavior is very high. Instead, we'll introduce a by-directional proxy object, DocView object between the Explorer and Office documents, which acts as a DocObject container to host Office documents and acts as a shell view object to be plugged into the right pane of the Explorer.

> *Notes: This was the least proven technology of all until a month ago, because of the complexity of three-party menu/toolbar negotiation. It, however, was mostly implemented and working well with Office 95 apps at this point. We should start testing this code (independently from the rest of code) very soon.*

BrowseContext and HyperLink (owned by SriniK -- Office):
Another new component in this mechanism is HLINK.DLL, which provides the implementation IHlinkBrowseContext and IHlink interface. This hyper-link interface layer (proposed by SriniK, Office group) is the mechanism that allows the browser (e.g., Explorer) to monitor all the hyper-link activation from the (DocObject) documents and create a per-browse context history (for back/next navigation).

> *Notes: This interface layer is the only new interface layer we introduce in Nashville/Office 96 and we are the first browser that uses this mechanism.*
>
> *Notes [Oct.27]: SriniK and I have agreed on how the shell will interact with hyper link interfaces. The DocView object will implement IHlinkFrame and it's Navigate method will appropriate convert monikers to pidls so that it fits in the explorer's name space. At this point, it is designed so that the shell does not directly load HLINK.DLL.*

Navigation UI in Explorer (owned by CheeChew -- Shell):
We are going to change the items on the Explorer toolbars and menus to make it easy to navigate (see ChristoB's document for detail). The work consists of two pieces, UI and hyper-link support. The UI work consists of navigation toolbar/menu and favorite/history folder.

> *Notes [Oct.27]: We have made a significant progress in October. We have decided to use a list of pidls (which specifies the location of document/folder in the shell's name space) to maintain the per-context navigation history instead of a list of IHlink objects (in IBrowseContext object). This straightforward mechanism allowed us to implement the navigation UI with a very small amount of code.*

DocObject support in HTML View (owned by JeremyS -- O'Hare):
The O'Hare team needs to re-package the existing Internet Explorer code so that the shell can "host" it in the right pane of the explorer. There are multiple choices (DocObject, ShellView, Window Control), but we have chosen to use DocObject mechanism to avoid putting any counter-part specific code in either side.

> *Notes: O'Hare team needs to make a significant changes to the existing code base, which is a monolithic exe, to make it thread-safe In-Proc server DLL. JeremyS estimate the cost to be 22 man-week. There are a few alternative and fall-back plans, but we can't get a realistic estimate until they start doing this work.*
>
> *Notes [Oct.27]: Single-instance version of HTML DocObject started working under Office binder. This is a great progress, but there are still many missing interfaces (such as IPersistFile, IHlinkSrc, etc.). It's clear that this work is going to be critical path of all the development work.*

Async URL Moniker (owned by SriniK/VictorS – O'Hare):
To allow Office apps to asynchronously render documents on WWW (actually Office 96 won't asnyc-render the main document, but it will delay render pictures), the O'Hare team needs to implement a async moniker which represents an URL connection and manage caches transparently.

Notes [Oct. 27]: Office 96 team (SriniK) has implemented their own URL moniker to meet their code complete date (good news). The ownership of this code will be transferred to O'Hare team soon and we hope that HTML DocObject will start using it very soon to find out all the issues.

HyperLink/Navigation support in Office 96 (owned by Office team)
This work is necessary to make Office documents act as hyper-link documents. In those documents, the user will see hyper links and the user can in-place navigate to the link sources simply by clicking those links. This work is not a required piece for Nashville release, but will just fill the last piece of jigsaw to complete the seamless navigation.

Notes [Oct. 27]: Although Office 95 documents don't allow the user to put hyper-links in them, the user will be able to in-place navigate to those document as leaf nodes.

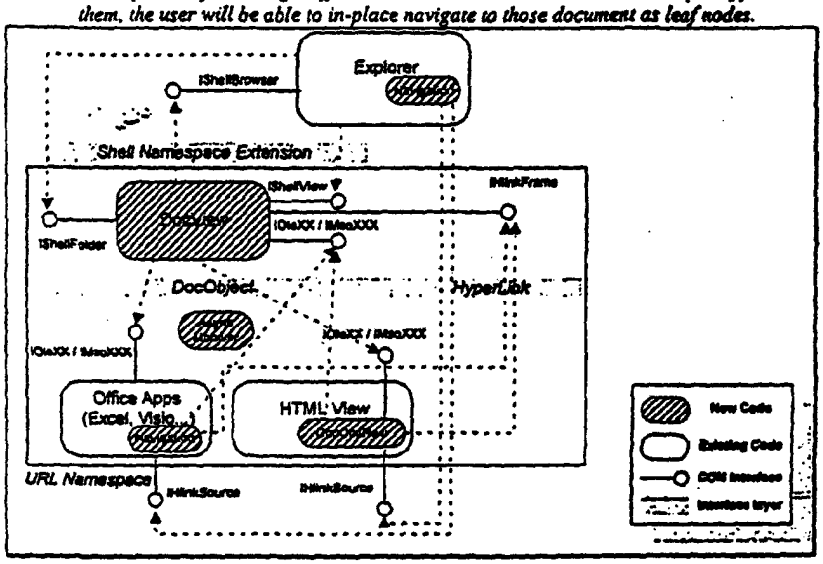


Fig.2 Shell Explorer - Internet Explorer Integration

MS9 000035
CONFIDENTIAL

MS98 0116192
CONFIDENTIAL

MS-PCA 1085018

MS-CCPMDL 000000226890
CONFIDENTIAL

Page View

Fig.3 and Fig.4 shows you how we will add a *page view* as the fifth view to the existing shell folder view (DefView) code (which has *icon*, *small-icon*, *list* and *detail* views). We will use the DocObject mechanism to host HTML view window within the shell folder window, because that is the interface the HTML viewer will export for the integration (described in the previous section). When we make this change to the shell folder view code, we will make it generic enough so that we (or even third party) can add a non-HTML viewer (such as RTF or Word document) as an additional view of the associated folder in future. There are three components to be either modified or creates:

DocObject hosting in Shell Folder View (owned by GeorgeP):
We need to change the Shell Folder View (DefView) code to support an additional view -- a page view -- which provides a single-click navigation, cool graphics and customization. We have chosen to use the HTML format (over RTF and Word document format) mainly because it is easy to either generate or pre-process HTML scripts.

> *Notes: As I mentioned in the Overview section, allowing third party to add extra non-HTML pages is not one of our goals in this release. It, however, does not prevent us from making it so, if it is the most appropriate mechanism to support "page" view.*
>
> *Notes [Oct.27]: It is getting clear which feature is easy to implement (single shell view) and which feature is difficult (multiple shell view, sharing selection state). ChristoB is finalizing the list of prioritized features based on this feedback. This is probably the area we should most carefully manage the feature/schedule trade-off.*

DocObject support in HTML Viewer (owned by JeremyS – O'Hare):
This is exactly the same piece of code we need for the integration of the Internet Explorer and the Shell Explorer.

OCX hosting in HTML Viewer (owned by PhilCo/VictorS – O'Hare)
This is the work O'Hare team will do anyway to compete against the other Web-browsers. "ShellFolderView OCX" section (below) describes how we use will it. To avoid creating temporary files, we need to come up with an extension to HTML to initialize each OCs with strings.

> *Notes [Oct.27]: This code started working. It allows us to create a HTML page which has a shell view embedding in it. Currently, there is no way to pass IStream to an OCX, which we need soon. The shell view will load various states (folder location, background color, view state, etc.) from that stream to initialize itself. We should make it sure that it is very efficient.*
>
> *Notes [Nov. 8]: We know exactly what we need. The HTML tag for an OC should allow us to give a initializing string, which will be passed to the OC via the Load memver of IPersistStreamInit. PhilCo owns this issue.*

UI Enhancement to HTML Viewer (owned by VictorS – O'hare)
We'd like to make some minor UI changes to the HTML viewer so that it looks cool and we can make our page view as function as the other views. It includes (1) hot link highlighting, (2) source context menu, (3) source data dragging, (4) BMP/ICO support and (5) command invocation support. GeorgeP has a complete list (already sent to VictorS).

HTML generation mechanism (owned by GeorgeP – Shell):
We'll generate a HTML file based on some template mechanism and pass it to HTML DocObject via IOleObject::InitFromData (instead of IPersistFile::Load to avoid creating a temporary file).

> *Notes [Oct.27]: VladS will take the ownership of peer-to-peer HTTP server work from me. He is going to look at my code and Gibraltor code and find out the best solution for Win96. He should at lest bring some security code from Gibraltor into our light-weight HTTP server and integrate it with the sharing UI of file sharing mechanism. I believe shipping this server (which supports BGI interface already) in Win96 is critical for the success of Gibraltor. This is probably the only way to convince multimedia developper to develop their multimedia titles as BGI DLLs.*

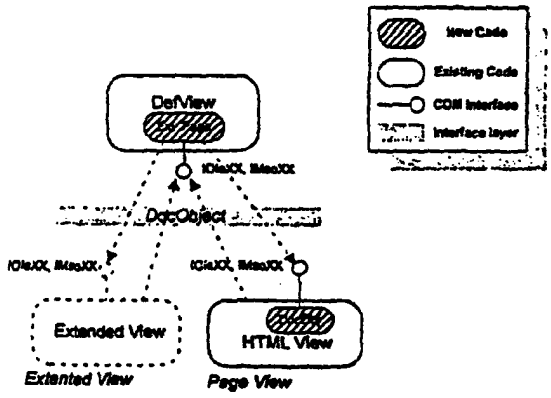
Notes [Nov.8] We significantly simplified the mechanism from the previous proposal by using non-file based initialization scheme.

ShellFolderView OCX (owned by SatoNa/GeorgeP – Shell)

One of the UI requirement of this page view is to have a fully functional folder view within the page view. To achieve this, we need to create a window in the HTML view window and be active all the time. Since our HTML view supports OCX hosting for another reason anyway and OCX provide us enough flexibility, we have chosen to make a ShellFolderView OCX. Another advantage of using OCX is component testing. Since there are already many stable OCX container code, we can easily test our code – which is much easier than testing with a new code (OCX hosting HTML view code).
We can either creates a OCX wrapper object on top of the existing ShellFolderView code or directly put a OCX support code in ShellFolderView. Although the latter approach would be efficient, we'll pursue the former approach to minimize the changes to existing code and cut some development cost (by reducing the complexity – the ShellFolderView code is already complex enough).

Notes: The ShellFolderView OCX will make it possible to embed it in arbitrary OLE container. For example, I can embed a ShellFolderView to my shared directory in my mail message and send it to you. When you open that message, the ShellFolderView OCX establish an appropriate UNC connection and fills contents.

Notes [Oct.27]: ShellFolder embedding work is done. We can embed it in WordPad, Excel or Exchange client very nicely (there is a minor problem with Word, which is supposed to be fixed when we fully support OCX interfaces).



MS9 000037
CONFIDENTIAL

MS98 0116194
CONFIDENTIAL

MS-PCA 1085020

MS-CCPMDL 000000226892
CONFIDENTIAL

Fig.3 Page-view Support

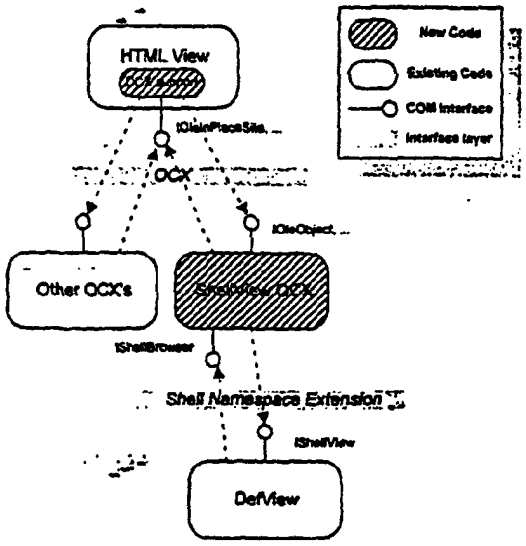


Fig.4 Embedding a ShellFolderView OCX in a HTML document