Clarion Magazine

Please note that Clarion Magazine is temporarily publishing three issues per month rather than four so we can devote more time to our upcoming book series. All subscriptions are automatically extended.

PDF For October, 2003

Articles: XML

News: XML

All Clarion Magazine articles for October, 2003 in PDF format.
Posted Monday, November 03, 2003

Weekly PDF For October 26-31, 2003

All articles for October 26-31, 2003 in PDF format.
Posted Monday, November 03, 2003

Clarion 6 Goes Gold!

Clarion 6 has gone gold! A final patch will be made available to EA program participants, and shipments will begin after CDs are received from the duplicators (which takes 8-10 business days).
Posted Monday, November 10, 2003

Clarion 6 First Look: The Templates

In Part 3 of this series Dave Harms looks at the changes in the template directory from C5.5 to C6.
Posted Thursday, November 13, 2003

The Windows API: Downloading Files Part 1

The Windows API offers numerous benefits to Clarion developers, including performance improvements and productivity gains from leveraging existing, reliable code. In this series Veronica Chapman demonstrates how to use the Windows API to connect to the internet and download files. Part 1 of 3.

News

Nice Touch Solutions Product-Wide C6 Gold Support

MAV Direct ODBC 0.03 Beta

xPictureBrowse 2.1

Clarion Third Party Profile Exchange Updated

Insight Graphing 1.26

EasyResizeAndSplit 2.04

BST Version 2.1

Clarionfoundry Preferences

MyFavoriteEmbeds Release 2.0

IngasoftPlus Products C6 Compatibility

SURVEY

What kind of backup system do you primarily use?

Tape

24.3%
CD

36.5%
DVD

8.1%
Other optical

0.0%
Hard drive

23.0%
Internet

2.7%
Other

5.4%
None

0.0%

74 responses

Previous Surveys

Posted Friday, November 14, 2003

Weekly PDF For November 9-15, 2003

All articles for November 9-15, 2003 in PDF format.

Posted Monday, November 17, 2003

Moving Applications to Oracle: RI And AutoNumbering Part 1

Jon Waterhouse takes a detailed look at the differences between relational integrity and autonumbering in Topspeed databases and Oracle databases. Although specifically about Oracle, this series also contains useful general information about converting to SQL. Part 1.

Posted Friday, November 21, 2003

The Windows API: Downloading Files Part 2

The Windows API offers numerous benefits to Clarion developers, including performance improvements and productivity gains from leveraging existing, reliable code. In this series Veronica Chapman demonstrates how to use the Windows API to connect to the internet and download (and optionally expand) files. Part 2 of 3.

Posted Friday, November 21, 2003

Weekly PDF For November 16-22, 2003

All articles for November 16-22, 2003 in PDF format.

Posted Tuesday, November 25, 2003

Moving Applications to Oracle: RI And AutoNumbering Part 2

Jon Waterhouse takes a detailed look at the differences between relational integrity and autonumbering in Topspeed databases and Oracle databases. Although specifically about Oracle, this series also contains useful general information about converting to SQL. Part 2.

Posted Friday, November 28, 2003

Native XP-Menu For C55 ABC And Legacy

Free Stuff For C6

NSA Guides To Network Security

Ace Icons Named Third-Party Vendor Of The Week

Compad's XPMenu Crossgrade To Poweroffice's XPMenu

Virtual EIP ABC Templates C6 Compatible

Nice Touch Solutions C6 Gold Support

Clarion Developers Challenge Week 11 Winners

LogFlash C6 Gold Compatible

MessageBox 1.75

xXPpopup v1.0

Favorite Embeds Template

Holiday Treats From Gitano Software

Gitano Software Utilities Update

Native XP Panel Source Code

One Year Ago In CM

PDF for November, 2002

Weekly PDF For November 24-30, 2002

PostgreSQL 101: Getting Started

Two Years Ago In CM

The Clarion Advisor: Sizing Windows

Using SQL Server's Data Transformation Services (DTS)

Calling By Address, STARTing By Address (Part 2)

Three Years Ago In CM

Profiler/Debugger Tools Update

Making Sense of ABC's ErrorClass - Part 2

Clarion News - November 2000

Four Years Ago In CM

ABC Design Series:

The Windows API: Downloading Files Part 3

The Windows API offers numerous benefits to Clarion developers, including performance improvements and productivity gains from leveraging existing, reliable code. In this series Veronica Chapman demonstrates how to use the Windows API to connect to the internet and download (and optionally expand) files. Conclusion.

Posted Friday, November 28, 2003

Looking for more? Check out the **site index**, or **search the back issues**. This site now contains more than 700 articles and a total of over a million words of Clarion-related information.

EasyExcel 3.03

CPCS v6.00 Gold Released

Clarion6 Gold Pre-Release Patch

All CapeSoft Accessories Ready For Clarion 6 Gold

BST Scheduler Resource Manager Suite

ImageEx 2 Anniversary Special

Fenix 1.3 Released

XP Taskpanel

ProDomus Software Named Third-party Vendor of the Week

Clarion Developers Challenge Week 10 Winners

EasyMultiTag 2.04

Professional Wizards Half Price

Easy3DStyle 1.05

ABC Free Templates Final Pre-Gold Beta

Flash Videos Demo C6 Features

Clarion/ASP 1.3 December Release

The ViewManager Part 1

Clarion News - November 1999

The Art Of Software Development: Analysis By Design

DevCon 2004

xWindowSettings 2.11b

XPMenu Updated

Clarion Essentials Course for Cape Town Rescheduled

EasyResizeAndSplit 2.03

xToolTip v1.5

Gradient Released

Free Clarion 6 Screen Saver

Pea Brain Software Named Third-party Vendor Of The Week

BMT Menu Template Released

Replicate 1.25

New CapeSoft NT Service Product Expected Nov 17

RADrace 2003: Fenix Finishes Second

Clarion Developers Challenge Week 9 Winner

Press Release Service

Promotion Offer From Ingasoftplus And EC Software

Clarion Developers And IT

Companies

Clarion Jobs Page Updated

EasyResizeAndSplit 2.02

CopyFlash 2.3 Released

EasyListPrint 1.07

EasyReport 1.02

Clarion News Group Archives (NNTP)

Clarion News Group Archives (NNTP)

File Manager Beta 3.25

xWindowSettings v2.11

CapeSoft Free Stuff Page

Object Writer Version 2

Safe Reader & Safe Writer Version 2

Capesoft Replicate

CapeSoft Progress Goes Gold

Taboga BarCode Library 1.1

Sterling Data Products At CR-6 Level

Berthume Software Named Third-party Vendor Of The Week

Insight Graphing Beta 1.24

Clarion Developers
Challenge Week 8 Winner

Search the news archive

Clarion Magazine

Clarion Books

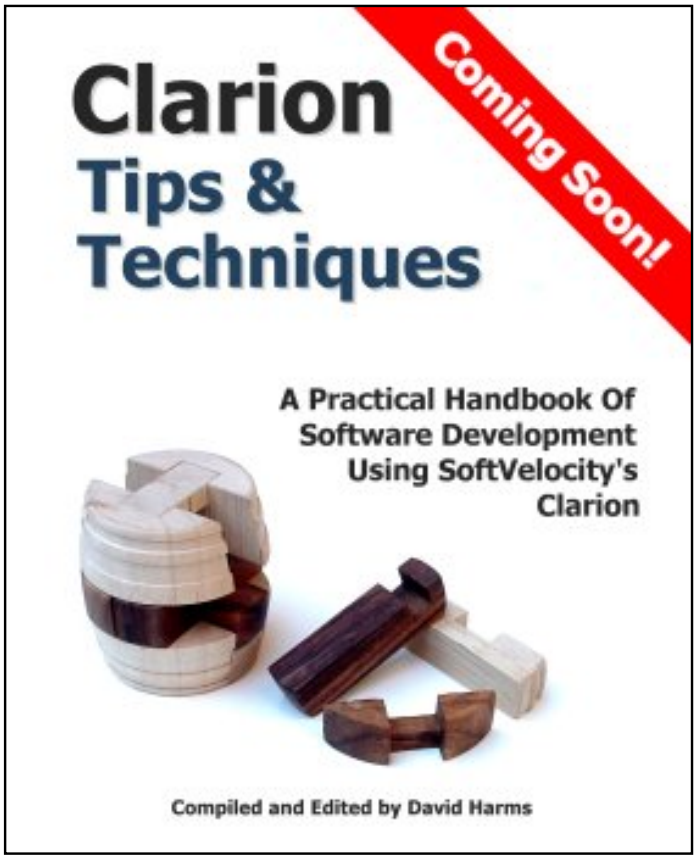
For Immediate Release: Clarion Magazine now has Clarion books in production

Although Clarion Magazine, the online publication, is very popular with Clarion developers, there is still something to be said for good old printed paper. And while some subscribers print the monthly PDFs, this still isn't quite the same thing as reading a real book.

I'm pleased to announce that we have started converting Clarion Magazine's HTML pages to book form. There is a stunning amount of material to choose from; in a 7.5" x 9.25" book format, the Clarion Magazine web site contains some 6500 pages of Clarion articles. That's about two feet of shelf space! If there is sufficient interest we will consider publishing all of this material, but it would be impossible to bring it all to press in a short period of time. As a result, we're focusing our efforts on areas likely to be of most immediate interest to Clarion developers.

Our first book: Clarion Tips & Techniques

The first Clarion Magazine book will be a Tips & Techniques volume of approximately 600 pages, containing selected articles from Clarion Magazine's first five years in publication, as well as a small number of articles originally published in Clarion Online. These articles were written for Clarion 5.0 and 5.5, but many are just as applicable to Clarion 6.



Check this page regularly for updates, including an upcoming table of contents and a PDF excerpt!

We'll be using a major US printing house to produce these perfect bound, soft cover books. The quality is the same high level as you would expect to find in any title from a mainstream computer book publisher.

The conversion of all HTML documents to a suitable book format is almost complete (mostly this involves making the HTML consistent from one document to the next, and then fine tuning the code that converts the HTML to book format). Once that task is done the Tips book will get a copy edit and final formatting. The next step is adding a full index. Once the indexing is under way and we have a rough idea of when the book will go to the printer, we'll begin taking orders. We expect the book to begin shipping in January.

Notify me when this book is available for purchase!

Email address: Notify Me!

You may also want to use our RSS news feed to track the book's progress. We'll be posting news items as the book's production schedule progresses.

Books to come

With 6500 pages of articles, we have a lot of topics to choose from. In the near future we'll be adding a survey to this page so you can let us know which topics interest you the most.

Check back regularly for updates! If you have any questions just email.

Clarion Magazine

Clarion 6 First Look: The Templates

by David Harms

Published 2003-11-13

In this third installment of my first look at C6 (which is has now been released), I'll continue my exploration of the changes to the support files with a look at the template directory. If you haven't already read the first and second installments, you may want to do so now.

Next, if you haven't already seen the new SoftVelocity web site, you should visit it. This is one of the most welcome changes accompanying C6. The new site has a new look (graphics by Jesus Moreno at Gitano Software), and contains a lot of information about C6, including some Flash videos highlighting key C6 features and enhancements. And at long last there is a price list! As of this writing, upgrade prices are as follows:

C6 Professional: $350
C6 Enterprise: $799

New licenses are:

C6 Professional: $799
C6 Enterprise: $2499

You can also get a competitive upgrade to the Professional edition for $350, with the instructions to contact SoftVelocity for information on qualifying products. There is a feature matrix that compares Enterprise and Professional. This list would seem to be incomplete, however, as it lists relatively few features as present in Enterprise only (such as Oracle Call Interface 8.x and BLOB support, the business graph class, and report output generators for PDF, XML, HTML, Text). Enterprise Edition also includes Data Modeller, the Business Math Library, the Dictionary Synchronizer, several SQL accelerator drivers, and version control support.

Legacy changes

If you're not yet familiar with Clarion's template language, just think of it as an interpreted

programming language, with the Application Generator (AppGen) as the interpreter. While some of the AppGen's functionality is hardwired, much of what you see in the AppGen, and all the code it generates, is dependent on the templates.

There are still two template sets (a.k.a. chains) shipping with C6. The Clarion (or Legacy) chain dates to the first version of Clarion for Windows. (Although SoftVelocity favors the term Clarion for this chain, I will call it Legacy here, simply to avoid the confusion of calling a Legacy application a "Clarion" application.) The Legacy templates generate only procedural code, at least up until C6. That means that *all* the non-embedded code in your Legacy app is generated by templates. In ABC, the templates generate largely object-oriented code, and make heavy use of the ABC class library. ABC templates typically declare objects that are instances of, or derived from, ABC classes.

SoftVelocity has chosen to continue support of the Legacy templates, and clearly this presents a problem: How do you maintain new features across both chains? SoftVelocity's answer is to implement new features for ABC, and where possible modify the Legacy templates to optionally make use of ABC classes. If you view the global properties of a C6 Legacy app, you'll see a new Classes tab, as shown in Figure 1.

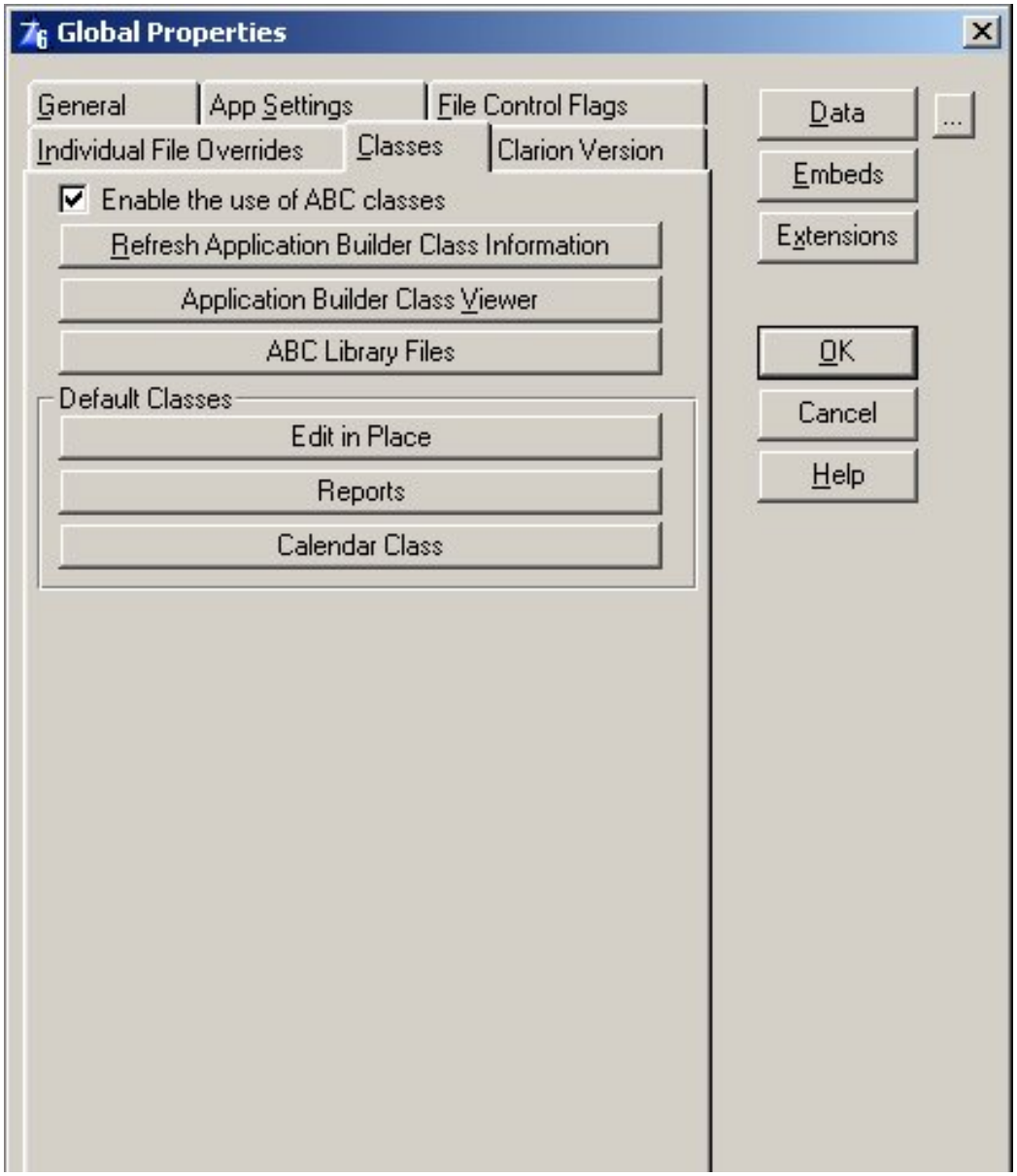


Figure 1. The Classes tab in a Legacy application

If you check the Enable use of ABC Classes button, you'll be able to take advantage of the new Edit-In-Place functionality, reporting enhancements, and the new Calendar class.

In fact, there are a number of areas where the Legacy and ABC templates now overlap. Figure 2 shows the App Settings tab for a Legacy app.

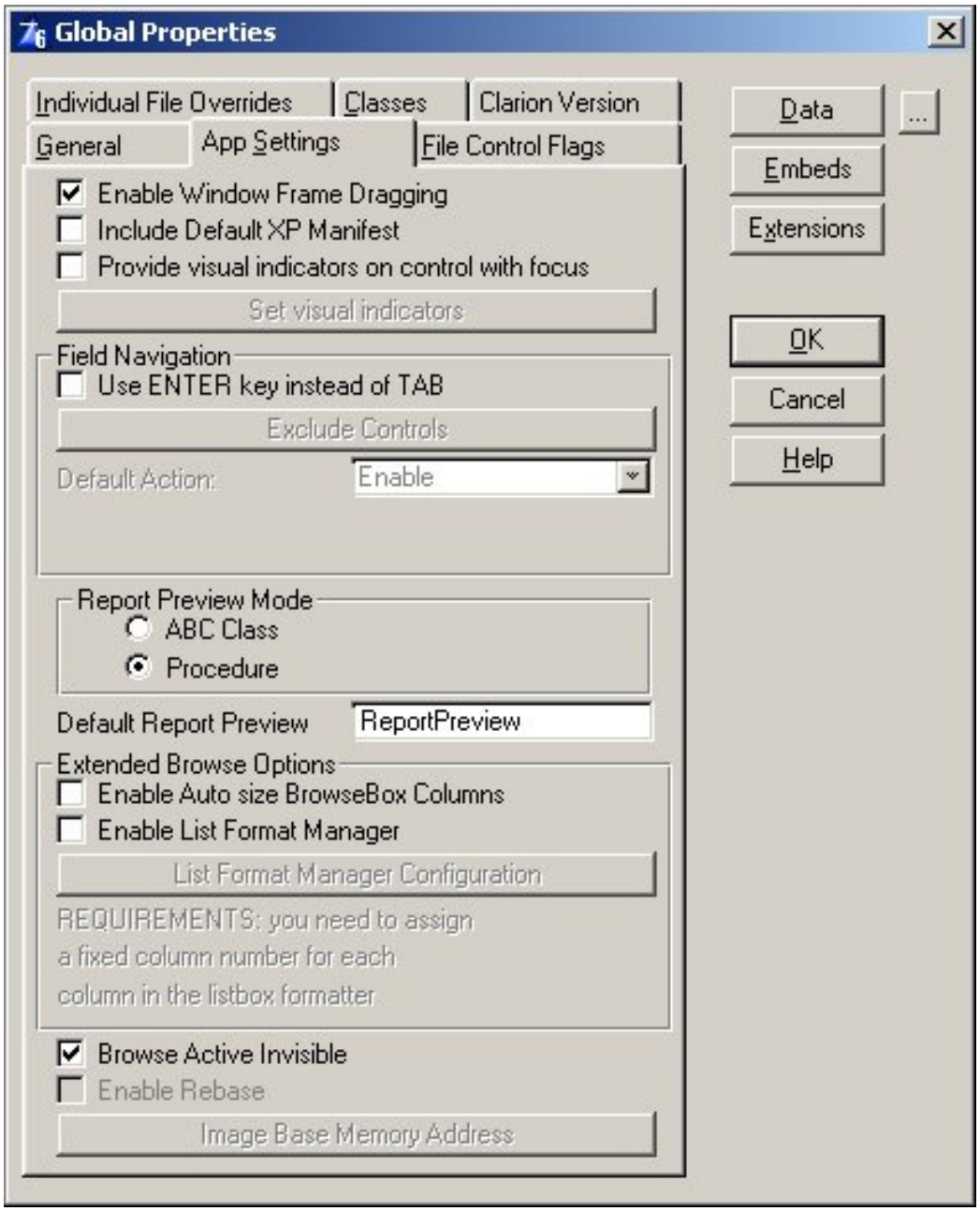


Figure 2. Legacy App Settings

The App Settings tab is the same in both Legacy and ABC apps, with the exception that ABC apps do not have the choice of ABC or procedure report preview mode. That means that Legacy apps can take advantage of new features such as:

- XP manifests, to give apps a native XP look and feel
- New visual indicators on forms to show field with focus – these include setting the field color, putting a box around the field, or showing an indicator character beside the field.
- Using the Enter key to navigate, instead of the Tab key
- Browse enhancements, including user-definable list formats (users can choose which columns to display, and in which order)

There's more, but I'm getting ahead of myself. It's time to examine the templates!

The templates

As in Parts 1 and 2, I'm using the directory comparison utility Beyond Compare to discover what's changed between Clarion 5.5 and Clarion 6. Before I hit the highlights, there are a few general points worth noting. First, there don't seem to be a whole lot of template changes related to the new threading model. As with the relatively low number of changes to the ABC library, I think this is a good omen for those worried about making the switch. Although the new threading library was a massive internal change, and one absolutely necessary for the language, it doesn't look like it will have a big impact on the average developer. (True threading does open up a whole lot of new opportunities, but that's not the subject of this article!)

As well, there are some minor enhancements sprinkled throughout the templates to make life a little bit easier. One is the addition of numerous prompts for the Expression Editor; another is the addition of file dialogs where previously you had to type the name of the file. ABC also includes improved support for additional sort fields on browses, processes and reports, and there is a new set of Higher Key Component prompts, which you can use when specifying a range limit component of a key that is *not* the highest component of the key. Use the prompts to specify what the higher component value(s) will be.

Now for some specifics:

ABBLDEXP.TPW – (ABC) Adds the new rebasing code.

ABBLOB.TPW – (ABC) The new `BlobInControl` extension template (ABC and Clarion chains) handles all of the code needed to display a BLOB field as a hot field for a browse or report, or on a form.

ABBROWSE.TPW – (ABC) The browse template now has the Report attribute, which means that the Report Formatter is available for browse procedures. There is a new Extended Options tab to manage the aforementioned user-defined list formats, and to enable click-on-column browse sorting. If you have colors enabled for your browse, you'll see the new greenbar option. If you're using an SQL database, you will also see a tab called SQL Advanced. This

lets you assign SQL expressions to fields in the view; you could use this to include expressions like count(*) or, presumably, a sub-select if the database supports it. You can also add `GROUP BY` and `HAVING` clauses. And there is a new BrowseNoRecordsButton control template, which as the name suggests is a button that is hidden or disabled when there are no records in the browse. You might use this to prevent the user from clicking on, say, a details button if there are no records.

ABCHAIN.TPL – (ABC) This master template file for the ABC chain adds support for saving INI data to the registry (INI File Settings becomes Non Volatile Storage Settings). There are a number of global options related to new features discussed elsewhere in this article, including list box formats, enhanced field focus, calendars and rebasing. There is also a new Clarion Version informational tab, which lists the Clarion version, Template Family, Template Version, and ABC Version.

ABCODE.TPW – (ABC) The `LookupNonRelatedRecord` code template lets you specify higher key components. The CallProcedureAsLookup code template supports passed parameters.

ABCNTRL.TPW – (ABC) The `DOSFileLookup` control template now lets you choose single or multiple files, and also directories instead of files. There is a default mask for image types, including JPG, BMP, and GIF.

ABDROPS.TPW – (ABC) The File Drop templates higher key component support, and can also have greenbars.

ABFILE.TPW – (ABC) This template now includes code for client side triggers.

ABMAIL.TPL – (ABC) This template adds SMTP authentication prompts, plus a large number of small coding changes

ABOOP.TPW – (ABC) This new file contains `#GROUP`s used by the ABC templates.

ABPROCS.TPW – (ABC) The Source procedure template now has file opening/closing code. The Process procedure template gets the REPORT attribute, which makes the Report Formatter available. Other changes include an option to make progress windows MDI. A process's data source can be a queue or string, as well as a file.

ABREPORT.TPW – (ABC) This file has, as you'd expect, a lot of changes to support all the new report output options, as well as the ability of reports, like processes, to now use queues and strings as data sources, as well as files. Also new is the `BreakManagerClass`, which is designed to give the developer more control over where breaks happen in reports.

ABTHREAD.TPW - (ABC) Includes the global `GlobalCooperativeThreading` extension, which basically makes your application function the same way a pre-C6 application does. There is also a PreemptiveProcedure extension which lets individual procedures in such an application function using C6's preemptive threading model.

ABUPDATE.TPW - (ABC) Now includes template support for the new calendar class.

ABVCRFRM.TPW - (ABC) New control templates to add VCR support for forms, where you can page through data on a form without having to go back to the browse.

ABWIZARD.TPL, ABW?????.TPW - (ABC) The wizard templates have undergone significant changes. There is a new Report Label Procedure wizard, and a new Theme Maintenance wizard. The Theme Maintenance Wizard (which you run as a utility) makes it quite easy to create your own themes, which you specify when using a wizard to create a new application or procedure.

ABWINDOW.TPW - (ABC) One of the changes here is the addition of code to support reports in windows, to go along with the enabling of the Report Formatter. The standard code won't actually do anything with your report, but it will generate all the code from the formatter. Other changes include support for list box styles.

ADO?????.TPW – (ABC, Legacy) Templates for ADO browses and forms, including control templates.

BUILDEXP.TPW – (Legacy) Contains the new rebasing code. See Carl Barnes' articles on rebasing for more on this subject.

CTLBROW.TPW – (Legacy) Support for new additional browse sort fields, range limit boundaries, selecting higher key components, greenbars, and conditional totaling.

CTLBROWA.TPW - (Legacy) New `BrowseViewButton` control template, `BrowseNoRecordsButton` control template, update buttons control template has greatly changed prompts.

CW.TPL - (Legacy) The Clarion template chain (known to many of us as the Legacy templates) now has the ability to use ABC classes, which makes it easier for SoftVelocity to support new features across both the ABC and Clarion template chains. Legacy apps now also have support for XP manifests and rebasing, better dialogs for things like icons, and liberal use of Expression Editor prompts.

CWADO.TPL - (Legacy) ADO support for the Clarion (Legacy) template chain.

CWBLOB.TPW - (Legacy) BlobInControl extension template.

CWHHABC.TPW – (ABC) Various improvements to the ABC help templates.

HELPUTIL.TPW – (ABC, Legacy) A utility template (`ListHLPids`) which lists help IDs.

CWRTF.TPW – (ABC, Legacy) The control template is deprecated in favor of the `TEXT` control using the RTF extension template.

CWUTIL.TPW – (ABC) Code templates in this new file include `GetOSVersion`, `GetFullDragSetting` and `SetFullDragSetting`.

ENHANCED.TPW – (ABC, Legacy) Template groups to support visual enhancements for the field currently having focus.

FIELD.TPW - (Legacy) Changes include prompts to specify threaded or non-threaded procedure calls and procedure parameters. You can also now assign multiple fields on field lookup.

PROCESS.TPW - (Legacy) Lots of changes to this procedure template,

QCENTER.TPW – (ABC, Legacy) `BrowseQBEList` control template

RTAR????.TP? - (Legacy) These files contain report output (target) templates for the Clarion template chain.

RTFCTL.TPW – (ABC, Legacy) The `RTFTextControl` template.

RULESMAN.TPL – (ABC) This file contains templates to manage the new C6 business rules objects.

SV?????.TP? – (ABC) The template files beginning with SV contain a variety of templates including business graphing, utility GROUPs (think of them as template functions), selected control enhancement, sort order buttons, and list control styles.

VERSIONRES.TPL - (ABC, Legacy) A template to include version resource information in your application.

XMLSPRT.TPW – (ABC, Legacy) XML importing/exporting/viewing templates.

That about does it for the source review of Clarion 6. I had hoped to touch on some of the IDE changes in this issue but I'm out of space, so it'll have to wait until next time.

David Harms is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with Ross Santos of Developing Clarion for Windows Applications, published by SAMS (1995). His most recent book is JSP, Servlets, and MySQL, published by HungryMinds Inc. (2001).

Reader Comments

Add a comment

Clarion Magazine

The Windows API: Downloading Files Part 1

by Veronica Chapman

Published 2003-11-14

In a previous article for Clarion Magazine I extolled the virtues of using Windows API function calls wherever possible. I gave number of reasons for doing so. First of all there is a slight performance gain. Second, there is an also slight reduction in application size. Thirdly there is a shift of responsibility for application functionality away from you, and onto Microsoft. But most significantly, using the Windows API offers greater flexibility, and more functionality, within your .APP.

Put simply, you can make your application look more professional, and provide similar functions to those offered by the big software vendors … whilst at the same time using a far better programming language than most of them use.

This series of articles serves as an example of what you, a Clarion Programmer, can gain from twelve (fairly well chosen) Windows API functions. You will find it interesting if you have ever wanted to download files of your choice from somewhere on the Internet, decompress them as necessary, and install them under your application's direct guidance. In other words, this article series describes how to use the Windows API to create an Internet-capable File Download Manager.

Overview

I propose to start by reviewing the Windows API functions that are required. The first six Windows API functions I will initially focus on are (in order of usage):

```
InternetOpen()
InternetConnect()
FTPOpenFile()
FTPGetFileSize()
InternetReadFile()
InternetCloseHandle()
```

I will, somewhat in passing, give a mention to `InternetAttemptConnect()` and `InternetCheckConnection()`, which I include in this article more for the sake on completeness than for any other reason. I will, however, discuss the usage of `FTPGetFile()`, used in place of `InternetReadFile()`, and the reasons for using the latter rather than the former.

There are several ways to download files across the Internet, but one of the oldest and still most common is the File Transfer Protocol, or FTP. Why do I propose to use FTP? The reason is that it will safely support *any format, treating all files in DOS-Binary mode.* Consequently the mechanisms I describe here will quite happily download and process files of any kind … including the binaries: EXE, DLL, LIB, HLP, CHM etc. Even though (for example) a Word .DOC can often consist of mainly readable-ASCII text, it will normally contain many (embedded) binary 'control' characters to apply formatting, and so on. These binary embeds will download under the File Transfer Protocol - whereas the major alternative, Hyper Text Transfer Protocol (HTTP), could misinterpret them – and thereby create an incorrect local copy.

Connecting to the Internet

At first sight, the business of connecting your application to the Internet looks complex. It is, in fact, very simple.

There are three Windows API functions that establish an Internet connection. The first is `InternetAttemptConnect()`. When called, it will check to see if there is an ongoing session at the time and, if there is, will seamlessly integrate your application's Internet-based operations into the stream. If there is no ongoing session at the time (i.e. no connection established), then `InternetAttemptConnect()` will bring up the regular Dial-Up Window, and wait for the connection to be established. This is, however, as far as it will take you.

The second API is `InternetCheckConnection()`, which performs the same task as `InternetAttemptConnect()`, but – once an open session is available – will try to establish the presence of a URL specified in one of its call parameters. It will return a success or failure condition, to define whether or not the URL was located.

The third connection method (and the most useful in general terms) is the API `InternetOpen()`. This is the one that I will describe in detail.

InternetOpen()

The function of `InternetOpen()` is to not only slot your application into an Internet stream (either into an ongoing session, or by means of a new session created through the Dial Up Window), but also to prepare the ground on which the required file transfer can be based.

> Please note (before reading on) that in this article all Clarion Prototype specifications are expressed in my `ASCIIsz` notation. I employ this throughout my WINAPIX.CLW collection of Windows API prototypes. My `ASCIIsz` Data Type is simply a re-definition of the standard Clarion `CSTRING` Data Type (i.e. if you wish, read `CSTRING` where you see `ASCIIsz`). I find that, by using `ASCIIsz`, it is easier to see the correlation between the Clarion version and the original C Version written in Hungarian Notation.
>
> The other aspect that I should mention is that all examples are quoted for the Win32 ASCII Environment – leading to the appropriate Windows API functions being called by name appended with an `A` suffix. For a Unicode Environment, a `W` suffix would take the place of the `A`.)

With the exception of `InternetAttemptConnect()` and InternetCheckConnection() (which have already been discussed), `InternetOpen()` *must be the first API function your application calls* when trying to gain access to the Internet. Subsequent code will need the handle that is returned, `hInternetOpen`, as you will see later.

The Prototype of `InternetOpen()` is:

```
InternetOpen(*ASCIIsz szAgent,|
             DWORD    dwAccessType,|
             *ASCIIsz szProxyServerName,|
             *ASCIIsz szProxyBypass,|
             DWORD    dwFlags),hInternetOpen,|
                               PASCAL,|
                               RAW,|
                               NAME('InternetOpenA')
```

`szAgent` is any name you wish to use, but is typically the name of your application, for example "My Internet-based Application". A firewall, for example, is likely to respond to the `InternetOpen()` attempt by asking if "My Internet-based Application" can have access to the Internet.

`dwAccessType` has four possible values that are largely designed to support Proxy Serving (where the Internet connection is *indirect*):

```
INTERNET_OPEN_TYPE_PRECONFIG EQUATE(0)
```

This value tells Windows to use the Registry to determine how to support the connection. Settings in the Registry will define whether or not a Proxy is necessary, and what parameters are to be used all possible cases. Consequently this is the value to use by default, because the Registry will always contain everything necessary for your application to connect to the Internet in all but the most abnormal circumstances.

```
INTERNET_OPEN_TYPE_DIRECT EQUATE(1)
```

This value is used to specify direct use of a nominated server (that is, a server other than the computer running the application) to support the connection. This is beyond the scope of this article.

```
INTERNET_OPEN_TYPE_PROXY EQUATE(3)
```

This value is used to specify a named proxy server to support the connection, which (again) is outside the scope of this particular article

```
INTERNET_OPEN_TYPE_PRECONFIG_WITH_NO_AUTOPROXY EQUATE(4)
```

This value is similar to `INTERNET_OPEN_TYPE_PRECONFIG` but, additionally, prevents the use of JavaScript, MS JScript, or an Internet Setup (INS) file providing a Startup Script. No such script is used in example given here, and so this value is not relevant.

szProxyServerName and szProxyBypass are only required if the connection is to take place through a proxy server. In order to keep this article reasonably simple, I assume that this software is to be run on a direct Internet connection (rather than via a LAN), and therefore I don't propose to delve deeply into proxy servers. Consequently these two call parameters are not (actually) used in this example, and are included simply for completeness. Nevertheless the format of szProxyServerName is <protocol>=<protocol>://<proxy_name>. So, for example, something like ftp=ftp://proxy_name:21 would define a proxy FTP server as proxy_name, via the standard FTP port (21); szProxyBypass can be used to modify the proxy server mechanism. dwFlags has, for InternetOpen(), three (actually only two … as you will see) possible values:

```
INTERNET_FLAG_ASYNC EQUATE(10000000h)
```

This value (and I quote) "makes only asynchronous requests on handles descended from the handle returned from this function".

Yes … OK Microsoft. Thanks for that.

An asynchronous request means that the request can be made, and Windows will process it *in its own good time*. Meanwhile your application would be free to perform other processing, and check back later to see if the request had been handled. Alternatively an *Internet hook* could be applied – such that your application is notified once the request has been processed. None of this is relevant to this example.

```
INTERNET_FLAG_FROM_CACHE  EQUATE(01000000h)
INTERNET_FLAG_OFFLINE     EQUATE(INTERNET_FLAG_FROM_CACHE)
```

This tells the connection *not* to make network requests. All entities are returned from the cache. If the requested item is not in the cache, a suitable error, such as ERROR_FILE_NOT_FOUND, is returned. Since this article is concerned with establishing and using a live Internet connection, this value is not relevant here.

Putting this all together – in order to make practical use for most purposes - is much less complicated than it may look. In the normal context, szAgent = your .APP's Name, dwAccessType = INTERNET_OPEN_TYPE_PRECONFIG (= 0), szProxyServerName and szProxyBypass are both = '', and dwFlags = 0;

These are suitable declarations:

```
hInternetOpen     hInternet                      ! (a LONG)
szAgent           ASCIIsz('My Internet-based Application')
dwAccessType      DWORD(INTERNET_OPEN_TYPE_PRECONFIG)
szProxyServerName ASCIIsz('')
szProxyBypass     ASCIIsz('')
dwFlags           DWORD(0)
```

This is a suitable API call:

```
hInternetOpen = InternetOpen(szAgent,|
```

```
                                dwAccessType,|
                                ProxyServerName,|
                                ProxyBypass,|
                                dwFlags)
```

If `InternetOpen()` returns zero, the open (connection) request could not be processed, and the entire operation comes to a grinding halt. This should only happen if the user has no modem, or no Dial-up Networking - or some major blockage that stands in the way of Internet Access. Or no computer.

Assuming that `InternetOpen()` succeeds, it will return a non-zero handle, known here as hInternetOpen. This is required by the InternetConnect() function, below.

InternetConnect()

The purpose of InternetConnect() is to establish a connection to the specific Web Site defined by a call parameter URL, under a specified transfer protocol. In other words InternetConnect() will attempt to log your application in to the server – in this case an FTP server (although not all Internet connections will require a user name and password).

To do this the user name and password must be available to your application in some way or other – either by hard coding, from an encrypted file, or from the Registry, etc. The Prototype of `InternetConnect()` is:

```
InternetConnect(hInternetOpen,|
                *ASCIIsz szServerName,|
                UINT     uServerPort,|
                *ASCIIsz szUserName,|
                *ASCIIsz szPassword,|
                DWORD    dwService,|
                DWORD    dwFlags,|
                *DWORD   dwContext),hInternetConnect,|
                                    PASCAL,|
                                    RAW,|
```

Once again, things are less fearsome than they might seem. `hInternetOpen` was returned by `InternetOpen()`. `szServerName` is the basic URL of the Web Site e.g. `ftp.ora.com.` `uServerPort` takes a value defined in conjunction with `dwService`, below. (Actually it is easier than that, as you will see). The possible values for `uServerPort` are:

```
INTERNET_INVALID_PORT_NUMBER EQUATE(0)
```

This value means "use the protocol-specific default port number" as implied by `dwService`. Consequently this value is a perfectly good choice for this parameter.

```
INTERNET_DEFAULT_FTP_PORT EQUATE(21)
```

This is the default port number for the File Transfer Protocol Service/Servers (FTP). This value could, of

course, be used instead of `INTERNET_INVALID_PORT_NUMBER` for an FTP-based session.

```
INTERNET_DEFAULT_GOPHER_PORT EQUATE(70)
```

This is the default port number for the Gopher Transfer Protocol (GOPHER)

```
INTERNET_DEFAULT_HTTP_PORT EQUATE(80)
```

This is the default port number for the (normal) Hyper Text Transfer Protocol Service/Servers (HTTP)

```
INTERNET_DEFAULT_HTTPS_PORT EQUATE(443)
```

This is the default port number for the Hyper Text Transfer Protocol Security Service/Servers (HTTPS)

```
INTERNET_DEFAULT_SOCKS_PORT EQUATE(1080)
```

This is the default port number for the Windows Sockets Firewall Service/Servers.

The best value for `uServerPort` is `INTERNET_INVALID_PORT_NUMBER`, which makes it dependent on the service chosen. `szUserName` is the user name that gains access to the Web Site. `szPassword` is the password that gains access to the Web Site. `dwService` is the means of specifying the transfer protocol … in this case FTP. The possible values of `dwService` are:

```
INTERNET_SERVICE_FTP     EQUATE(1)
INTERNET_SERVICE_GOPHER  EQUATE(2)
INTERNET_SERVICE_HTTP    EQUATE(3)
```

Therefore, little more needs to be said – except that by making `uServerPort = INTERNET_INVALID_PORT_NUMBER`, Windows will work out which port to use, in this case on the basis of `dwService = INTERNET_SERVICE_FTP`. `dwFlags` could take a number of possible values at this point, but they are for special circumstances, and not relevant within the scope of this article. For more detailed information consult the Microsoft Libraries. In this context `dwFlags = 0` is the appropriate choice (i.e. nothing special). `dwContext` is an application-specific code that can be used to signify (to, for example, the Asynchronous Notifications discussed earlier in the context of `INTERNET_FLAG_ASYNC`) anything you need to signify. This facility is not used by the example that accompanies this article, and therefore the value of dwContext is set to 0.

Putting this all together is, once again, not very complicated. It simply boils down to `hInternetOpen` being returned from the call to `InternetOpen()`, `szServerName = *web site url*`, `uServerPort = INTERNET_INVALID_PORT_NUMBER`, `szUserName = *username*`, `szPassword = *password*`, `dwService = INTERNET_SERVICE_FTP`, `dwFlags = 0`, and `dwContext = 0`:

Here's some sample data:

```
hInternetConnect  hInternet                    ! (a LONG)
szServerName      ASCIIsz('ftp.ora.com') ! (e.g.)
```

```
uServerPort       UNIT(INTERNET_INVALID_PORT_NUMBER)
szUserName        ASCIIsz('username')           ! (e.g.)
szPassword        ASCIIsz('password')           ! (e.g.)
dwService         DWORD(INTERNET_SERVICE_FTP)
dwFlags           DWORD(0)
dwContext         DWORD(0)
```

And here's the function call:

```
hInternetConnect = InternetConnect(hInternetOpen,|
                                   szServerName,|
                                   uServerPort,|
                                   szUserName,|
                                   szPassword,|
                                   dwService,|
                                   dwFlags,|
                                   dwContext)
```

If InternetConnect() succeeds it will return a non-zero connection handle, and will have established 'logged-in' FTP access into the Web Site. This connection handle, hInternetConnect, is then passed on to FTPOpenFile().

FTPOpenFile()

The purpose of FTPOpenFile is to navigate through the web site to the precise file required, and then to open it – in this case in read mode.

The Prototype of FTPOpenFile() is:

```
FTPOpenFile(hInternetConnect,|
            *ASCIIsz szFileName,|
            DWORD    dwAccessMode,|
            DWORD    dwFlags,|
            *DWORD   dwContext),hFile,|
                             PASCAL,|
                             RAW,|
                             NAME('FtpOpenFileA')
```

`szFileName` defines the navigation. It must include all folder paths, the file's name, and the file's extension (as applicable). Depending (usually) on the server's operating system, the file name may be case sensitive. An example of `szFileName` (quoting one of my previous articles for Clarion Magazine) is `'/cmag/v5/v5n08truth.html'`. `dwAccessMode` is used to indicate the file is to be opened in read mode (as compared to write mode). The appropriate value here is:

```
GENERIC_READ EQUATE(080000000h)
```

The same descriptions for `dwFlags` and `dwContext` apply here as given above for `InternetConnect()`. `dwFlags` has special values for special circumstances, and the `dwContext`

facility is unnecessary here, and consequently is not being used.

Therefore the usage of `FTPOpenFile()` condenses down to `hInternetConnect` having been returned from `InternetConnect()`, `szFileName` = the folder/path/filename.extension, `dwAccessMode = GENERIC_READ`, `dwFlags = 0` and d`wContext = 0.`

Here's the declaration:

```
FTPhFile            hFile               ! (a LONG)
szFileName          ASCIIsz('folder1/folder2/../file.ext')
dwAccessMode        DWORD(GENERIC_READ)
dwFlags             DWORD(0)
dwContext           DWORD(0)
```

And here is the function call:

```
FTPhFile = FTPOpenFile(hInternetConnect,|
                       szFileName,|
                       dwAccessMode,|
                       dwFlags,|
                       dwContext)
```

FTPOpenFile() will return zero if the file could not be located, otherwise it will return a non-zero, valid, file handle called – in this case - FTPhFile. This handle can then be used to drive first FTPGetFileSize()and then either InternetReadFile() or `FTPGetFilet()`.

FTPGetFileSize()

The purpose of this API call is to return the size, in bytes, of the file. The size returned is the exact storage requirement that will be required, and is consistent with the "number of bytes read" information returned by InternetReadFile().

The point here is that, in order to move information across the Internet (or any other medium), additional (control) bytes will prefix and suffix the actual data transferred. This will be determined by the transfer protocol.

In the case of FTP, for example, in order to transfer 1Mb of file data, approximately 1.4Mb is actually transmitted/received. This extra .4Mb is the transfer protocol overhead.

The transfer protocol overhead is not included within the information returned by FTPGetFileSize(), nor is it included in the information returned by InternetReadFile(). It does, however, show up on the Connection Status Window of an Internet or LAN connection. Consequently you will *not* be able to match up the value returned by FTPGetFileSize() with what you see on the Internet Connection Status Window.

The Prototype of FTPGetFileSize() is:

```
FTPGetFileSize(hFile,|
```

```
*DWORD dwFileSizeHigh),dwFileSizeLow,|
                       PASCAL,|
                       RAW
```

`hfile` specifies the file for which your APP requires the information. This was returned, in the form of `FTPhFile`, by `FTPOpenFile()`. `dwFileSizeHigh` is *returned* by `FTPGetFileSize()`, and so is `dwFileSizeLow`. Together they constitute a 64-bit file size, measured in bytes. Here's the data:

```
dwFileSizeHigh LONG
dwFileSizeLow  LONG
```

And here's the function call:

```
dwFileSizeLow = FTPGetFileSize(FTPhFile,dwFileSizeHigh)
```

Your application will need to process this information collectively, i.e. the high and low need to be merged together for practical usage. The simplest method is to convert to an `SREAL`, viz:

```
FileSizeBytes SREAL
```

taken with:

```
FileSizeBytes = ((dwFileSizeHigh * 65536) + dwFileSizeLow)
```

will produce the appropriate overall value.

InternetReadFile()

Earlier I stated that I would discuss the usage of `FTPGetFilet()`, used in place of `InternetReadFile()`, and the reasons for using the latter rather than the former.

`FTPGetFilet()`, taking `FTPhFile` as its handle parameter, could be used once all the above preliminaries have been made. However this API will download the entire file, *from end to end,* as a *stream*. In order to maintain some form of progress display for the user, it would be necessary to apply a suitable Internet hook, to drive an API called `InternetStatusCallback()`. Within `InternetStatusCallback()` it would be possible to compute, and display, any form of progress.

A simpler method, which does not require anyhooks/callbacks, is to employ `InternetReadFile()` for the job instead. The reason why this is simpler, is because you can tell `InternetReadFile()` how much of the file you want, wait for it to download that segment, and then tell it to read some more. And so on until the entire file has been downloaded.

By choosing the segments sensibly, and by driving the call to `InternetReadFile()` from the Clarion Window `EVENT:Timer`, your APP can perform the download, under control, and create any desired progress statistics for your user. Furthermore this latter method enables you to include a Cancel button, in the event that the file is very large, and your user may not wish to continue. `InternetReadFile()` will always return either a segment of the file, or an error code. If an error occurs, there may be an

additional delay whilst line time-outs and retries are exhausted. If the user decides to abandon the download during this short period, the effect of the cancellation may be a slightly delayed. In short, the Cancel button will respond as efficiently as any in this situation..

While this latter method is slightly slower than the streamed method used by `FTPGetFilet()`, you will find that it still operates with perfectly acceptable performance.

The Prototype of `InternetReadFile()` is:

```
InternetReadFile(hFile,|
                 *ASCIIsz szBuffer,|
                 DWORD    dwNumberOfBytesToRead,|
                 *DWORD   dwAmountDownloaded),BOOL,|
                                              PASCAL,|
                                              RAW
```

Here's the required data:

```
BlockSize               EQUATE(1024)
!
dwAmountDownloaded      LONG
dwNumberOfBytesToRead   LONG(BlockSize)
!
szBuffer                ASCIIsz(BlockSize + 1)
FileBinaryData          GROUP,OVER(szBuffer)
                           BYTE,DIM(BlockSize)
                        END
```

Note that four of these declarations are driven from the `BlockSize` equate, so they always remain synchronized. I chose the block (or segment) size as 1Kb as an example for average usage. It could easily be changed by adjusting the value of `BlockSize`. However, for simplicity, I recommend keeping to multiples of 256 – unless particular circumstances dictate otherwise.

I draw your attention to two other aspects. First, the size of `szBuffer` is defined as the block size plus 1. This is to allow for the extra zero-terminator byte that must always be available at the end of a `CSTRING`. Second, I draw your attention to the redefinition of the data space of `szBuffer` (by means of the `OVER`) as the `GROUP` named `FileBinaryData`. This is a simple construct to enable the 1Kb byte array (`BYTE,DIM(BlockSize)`) to be collectively moved to the `RECORD` contents of a local storage file.

Here's an abridged example of the code in action:

```
IF InternetReadFile(FTPhFile,|
                    szBuffer,|
                    dwNumberOfBytesToRead,|
                    dwAmountDownloaded) = True
  ... process the contents of szBuffer (as FileBinaryData) - up
      to the limit defined in the returned variable
          dwAmountDownloaded. For example:
```

```
  DOSFile:RECORD = FileBinaryData   ! ) Move to local storage
  APPEND(DOSFile,dwAmountDownloaded)
ELSE
  ... the download is complete
END
```

Note that `InternetReadFile()` will attempt to download the amount demanded by the `dwNumberOfBytesToRead` call parameter. While there is more than this amount left to download, it will return that same amount in `dwAmountDownloaded`. Towards the end of the file, i.e. once the `BlockSize` amount is less than what remains to be downloaded, the final segment will be downloaded, and the value returned in dwAmountDownloaded will reduce accordingly.

After the end of the file, `InternetReadFile()` will return `False` to indicate that there is no more to download, and `dwAmountDownloaded` will be returned as 0. Consequently, `dwAmountDownloaded` is a value that can be used directly in the DOS File `APPEND` statement – as shown above. `dwAmountDownloaded` is also the appropriate value to use when computing progress statistics, e.g. if `TotalAmountDownloaded` was intialized to 0 at the time `FileSizeBytes` was computed, then this code yields a progress percentage:

```
TotalAmountDownloaded += dwAmountDownloaded
ProgressPercentage = ((TotalAmountDownloaded|
                     / FileSizeBytes)|
                       * 100)
```

The call to `InternetReadFile()` is driven from the `EVENT:Timer` of a small Window that constitutes the File Download Management Function. In the appended example, I use a timing value of 100 mS, i.e. 10 segments per second. This can be tuned to taste.

Remember: `InternetReadFile()` does not return to your application's `ACCEPT` loop until the bytes have been moved over the Internet. Consequently, if you find that other processes noticeably slow down while File Download Management is in progress, then you can adjust your application in two ways:

a. Increase or decrease the value of `BlockSize`. – the direction to choose depends on the `EVENT:Timer` setting,
b. Increase the `EVENT:Timer` setting (i.e. fewer `EVENT:Timers` per second).

InternetCloseHandle()

The `InternetCloseHandle` function is simply used to close all of the handles created by the download process.

The Prototype of `InternetCloseHandle()` is:

```
InternetCloseHandle(hInternet),BOOL,|
                               PASCAL,|
                               RAW
```

Though not necessarily obvious, `hInternet` includes `FTPhFile`. This function returns `True` if the handle was closed successfully, and `False` otherwise. So what can you, the Clarion Programmer, do if it InternetCloseHandle() returns False?

Nothing. Nothing whatsoever.

If all other mechanisms have worked successfully, then any False return from InternetCloseHandle() is a Windows (i.e. Microsoft) problem – not yours. Therefore I tend to simply throw these conditions away, by defining an ever-present Global called WBR (Windows Boolean Return) = BOOL.

This allows me to code (in the appropriate place):

```
WBR = InternetCloseHandle(hInternetOpen)
WBR = InternetCloseHandle(hInternetConnect)
WBR = InternetCloseHandle(FTPhFile)
```

… and not even give the situation a second thought.

Summary of Part 1

It may be that you found the detail of this article a little hard to read - it is quite a lot to swallow in one gulp. To make the code easier to follow, the following is a shortened version of the code, which for simplicity excludes all tests for error conditions. Use this only to further your understanding, not as a basis for your own code (for that you can use the code in the downloadable source).

```
!Globals:
WBR                     BOOL  ! (Windows Boolean Return)
DOSFileName             STRING(300),STATIC

Locals & Code Statements:
BlockSize               EQUATE(1024)
!
hInternetOpen           hInternet
hInternetConnect        hInternet
FTPhFile                hFile
!
szProxyName             ASCIIsz('')
szProxyBypass           ASCIIsz('')
!
szAgent                 ASCIIsz(101)
szServerName            ASCIIsz(301)
szUserName              ASCIIsz(31)
szPassword              ASCIIsz(31)
szRemoteFile            ASCIIsz(301)
!
dwContext               DWORD(0)
!
szBuffer                ASCIIsz(BlockSize + 1)
```

```
  FileBinaryData           GROUP,OVER(szBuffer)
                              BYTE,DIM(BlockSize)
                           END
  !
  ProgressPercentage       LONG
  !
  dwFileSizeHigh           LONG
  dwFileSizeLow            LONG
  FileSizeBytes            SREAL
  !
  AmountDownloaded        LONG
  TotalAmountDownloaded    SREAL
  !
  ! DOS (Temporary) File:
  !
  DOSFile            FILE,DRIVER('DOS'),NAME(DOSFileName),PRE(DOS),CREATE
  Record                RECORD
  BinaryData               GROUP
                              BYTE,DIM(BlockSize)
                           END
                        END
  !
  ! Download Manager Window:
  !
  Window      WINDOW(' '),|
                     AT(,,378,38),|
                     FONT('MS Sans Serif',|
                     8,|
                     ,|
                     FONT:regular),|
                     CENTER,|
                     GRAY
                BUTTON('&Cancel'),|
                        AT(1,1,35,30),|
                        USE(?CancelButton)
                PROGRESS,|
                        USE(ProgressPercentage),|
                        AT(45,11,319,8),|
                        RANGE(0,|
                              100)
              END
   CODE
  !
   DOSFileName = 'local\path\of\folders\file.ext'
   CREATE(DOSFile)
   OPEN(DOSFile,1h)
  !
   OPEN(Window)
```

```
 Window{PROP:Text} = 'Connecting ... '
 SETCURSOR(CURSOR:Busy)
 DISPLAY()
!
 szAgent = 'My Internet-based Application'
 szServerName = 'ftp.yourwebsite.com'
 szUserName = 'webmasterusername'
 szPassword = 'webmasterpassword'
!
 szRemoteFile = 'remote/path/of/folders/file.ext'
!
 szProxyName = ''
 szProxyBypass = ''
!
 hInternetOpen = InternetOpen(szAgent,|
                         INTERNET_OPEN_TYPE_PRECONFIG,|
                              szProxyName,|
                              szProxyBypass,|
                              0)              ! (dwFlags)
 IF hInternetOpen NOT = 0
    hInternetConnect = InternetConnect(hInternetOpen,|
                                       szServerName,|
                          INTERNET_INVALID_PORT_NUMBER,|
                                       szUserName,|
                                       szPassword,|
                          INTERNET_SERVICE_FTP,|
                                       0,|     ! (dwFlags)
                                       dwContext)
    IF hInternetConnect NOT = 0
       FTPhFile = FTPOpenFile(hInternetConnect,|
                              szRemoteFile,|
                              GENERIC_READ,|
                              0,|             ! (dwFlags)
                              dwContext)
       IF FTPhFile NOT = 0
          dwFileSizeLow = FTPGetFileSize(FTPhFile,|
                                         dwFileSizeHigh)
          FileSizeBytes = ((dwFileSizeHigh|
                            * 65536)|
                             + dwFileSizeLow)
          TotalAmountDownloaded = 0
          Window{PROP:Text} = 'Downloading: '|
                             & CLIP(szRemoteFile)
          Window{PROP:Timer} = 10
          ACCEPT
             IF EVENT() = EVENT:Timer
                IF InternetReadFile(FTPhFile,|
                                    SzBuffer,|
                                    BlockSize,|
```

```
                                    AmountDownloaded)|
                                        = False|
                 OR AmountDownloaded = 0
                   SETCURSOR()
                   BREAK
                ELSE
                   DOS:BinaryData = FileBinaryData
                   APPEND(DOSFile,|
                          AmountDownloaded)
                   TotalAmountDownloaded|
                               += AmountDownloaded
                   ProgressPercentage|
                               = ((TotalAmountDownloaded|
                                  / FileSizeBytes)|
                                   * 100)
                   DISPLAY(?ProgressPercentage)
                END
             END
!
             CASE ACCEPTED()
             OF ?CancelButton
                SETCURSOR()
                IF MESSAGE(|
     'Are you sure you wish to Cancel this Download?',|
                           szAgent,|
                           ICON:Question,|
                           BUTTON:Yes+BUTTON:No,|
                           BUTTON:No) = BUTTON:Yes
                   BREAK
                END
                SETCURSOR(CURSOR:Busy)
             END
!
          END
          WBR = InternetCloseHandle(FTPhFile)
       END
       WBR = InternetCloseHandle(hInternetConnect)
    END
    SETCURSOR()
    WBR = InternetCloseHandle(hInternetOpen)
 END
!
 CLOSE(DOSFile)
!
 CLOSE(Window)
!
 RETURN
```

Conclusion to Part 1

In the second part of this article I will discuss how to process the file that has been downloaded in the event that it was compressed and needs to be expanded. The Source and .EXE are both included, in the downloadable file, but to be able to compile it yourself you will also need to download my collection of Windows API Prototypes (WINAPIX.CLW).

Download the source

Download WINAPIX.CLW

Download the source from www.merlinto.com

Download WINAPIX.CLW from www.merlinto.com

Veronica Chapman earned a B.Tech in Electronics & Electrical Engineering from Brunel University in 1968, and subsequently embarked on a career as a Programmer/Analyst, first writing code at machine level, and shortly thereafter working with real time systems and communications. By the mid '70s she was using languages such as COBOL and FORTRAN. Never a fan of BASIC or the C language, she discovered Clarion in the mid-90s and, ever since, has used it to create applications for the 16-bit (Win 3.x) and 32-bit Windows Platforms. An assembly code programmer from way back, Veronica discovered a cornucopia of very useful functions in the Windows API, and set about making these functions available to Clarion applications.

Reader Comments

Add a comment

It seems that there are Readers who (unfortunately) are...

Clarion Magazine

Clarion News

Search the news archive

Nice Touch Solutions Product-Wide C6 Gold Support

Nice Touch Solutions has announce Clarion 6 GOLD support for its entire line of Clarion add-on products. Clarion 6 support for Query, View, Report, Spreadsheet, and CrossTab Wizards includes both the ABC and Legacy Template chains for all five products.
Posted Tuesday, November 25, 2003

MAV Direct ODBC 0.03 Beta

MAV Direct ODBC 0.03 Beta is now available. Changes include: Fixed bug with composition of WHERE clause using Clarion LONG date and SQL DATETIME. MAV Direct ODBC is $99 during beta testing, $249 when it goes gold. Free upgrade for beta testers.
Posted Tuesday, November 25, 2003

xPictureBrowse 2.1

SealSoft's xPictureBrowse 2.1 is now available. Changes include: C6 (release candidate) support; Translation via TRN file; New xViewPictureForReport tepmlate to show picture from database; Minor bug fixes; New CHM help file. Registered users note that the install password has changed and will be emailed to you.
Posted Tuesday, November 25, 2003

Clarion Third Party Profile Exchange Updated

The Clarion Third Party Profile Exchange consists primarily of profiles of third party add-on products and vendors. This includes freeware templates and tools as well. Online and Downloadable Profiles available. Online product profiles include Product Internet URL, Order URL, Currency code, Dated Price Quote, Grouped by Category, Clarion 6 Compatible, Extended Description and Download Page Reference. Currently, there are 447 product profiles and 425 vendor profiles. You must have Product Scope 32 PRO Version 5.0 to view profiles with data files (downloadable profiles). 161 Clarion 6 compatible products as of this release.
Posted Tuesday, November 25, 2003

Insight Graphing 1.26

A new build of Insight Graphing is now available. Changes include: Hi-Lo graph: Pies restored to their full glory; Tooltips issues (hopefully) resolved; A new method for adding specific axis points; Support for Windows DPI settings.
Posted Tuesday, November 25, 2003

EasyResizeAndSplit 2.04

EasyResizeAndSplit 2.04 is now available. Changes include: New multi splits alignment strategiries; Changed embed points; Bug fixes.
Posted Tuesday, November 25, 2003

BST Version 2.1

New in BST version 2.1: C55-C6, Legacy and ABC compatibility; Enhanced runtime controls; New controls for calendars and weekly schedule; New embeds. This is a free update for registered users.
Posted Tuesday, November 25, 2003

Clarionfoundry Preferences

There is a new section on Clarionfoundry where you can set your preferences, such as your password and email address. You can also provide an alternative name/email address to supply for your published comments. You can also set default settings such as copyright/disclaimer statements for your posts, and much more.
Posted Tuesday, November 25, 2003

MyFavoriteEmbeds Release 2.0

The BFET My Favorite Embeds Template has been updated. New in version 2.0: Added %WindowEventHandling to embed buttons; Added Lists tab for ABC Browse boxes; Added %BrowserMethodCodeSection embed button; Added %ControlEventHandling for selected list box.
Posted Tuesday, November 25, 2003

IngasoftPlus Products C6 Compatibility

Ingasoftplus's EasyAnimation, EasyMultiTag, Easy3DStyle, EasyAutoEntry and EasyListPrint are now ready for C6 Gold.
Posted Tuesday, November 25, 2003

Native XP-Menu For C55 ABC And Legacy

Trond Eirik Paulsen's native XP-Taskpanel is now C55 ABC and Legacy compatible.
Posted Tuesday, November 25, 2003

Free Stuff For C6

The following are freely available from Gitano Software: C6 Screen Saver; Four C6 wallpapers; Two C6 icons, one transparent and one not, in 128x128, 48x48, 32x32, 24x24 and 16x16 in XP, 16 mil, and 256 format. (128x128 available in XP format only); Application icons for C6 applications; A group of icons to replace the standard toolbar icons in your applications - these icons are 16x16 and will not look good when used with previous Clarion versions (available in XP, 16 mil and 256 formats).
Posted Tuesday, November 25, 2003

NSA Guides To Network Security

Carl Barnes points out this National Security Agency document with useful information on

securing networks.
Posted Thursday, November 20, 2003

Ace Icons Named Third-Party Vendor Of The Week

Sue Pichotta and Ace Icons have been named by the Clarion Developers Challenge Football Contest as the Third-party Vendor of the Week. Sue Pichotta and Ace Icons are awarding this week's winner of the Clarion Developers Challenge Football Contest a copy of their Ace Icons Super Suite. Ace Icons has icon sets, button sets, and background sets for software and web developers. They also have several suites of icons, buttons and backgrounds. One of these is their Super Suite, which Sue and Ace Icons will be awarding to this week's winner of the Clarion Developers Challenge. For further information about these and other fine products from Ace Icons please go to http://www.aceicons.com.
Posted Thursday, November 20, 2003

Compad's XPMenu Crossgrade To Poweroffice's XPMenu

About two years ago, Ronald van Raaphorst created the XPMenu product, at first for his own use, then later as a third party tool. As XPMenu is completely written as a Clarion class, it has some limitations due to the fact that Clarion's listbox has some limitations. Now Trond Eirik Paulsen has created a better XPMenu (http://www.poweroffice.no/clarion/xptask.htm), which is also capable of showing more than one XPMenu on the same window, and has less flicker than Rondald's XPMenu. It also has the capability of using icons in the menu header, and a highlighting capability when hovering over a menu or an item. As Ronald's business goal is not primary to sell third party tools, but to sell applications to end-users, he has decided to stop the sales and development of XPMenu, and to start using Trond Eirik's XPMenu in new applications. All of Ronald's XPMenu customers will receive an email offering a crossgrade Trond Eirik's XPMenu.
Posted Thursday, November 20, 2003

Virtual EIP ABC Templates C6 Compatible

A Clarion 6 gold compatible release of the Virtual EIP ABC Templates has been posted to the usual update site. From within the Clarion IDE, choose Accessories > Help > VirtualEIP. Then click on "Documentation > Updates" for the download link.
Posted Thursday, November 20, 2003

Nice Touch Solutions C6 Gold Support

Nice Touch Solutions, Inc. is pleased to announce Clarion 6 support for its full line of Clarion add-on products which includes Query, View, Report, Spreadsheet, and CrossTab Wizards.
Posted Thursday, November 20, 2003

Clarion Developers Challenge Week 11 Winners

First place for Week 11 of the Clarion Developers Challenge goes to Bo Schmitz. It was a tough week for all competitors, but Bo pulled away from the pack with an amazing 12 correct picks, while Alejandro Contreras, Dave Beggs and Jerry Davis were second with 10 correct picks. Congratulations, Bo! For finishing first in this week's Clarion Developers Challenge, Bo will be receiving a copy of The PD "Universal" Drop Edit Controls from Phil Will and ProDomus Software. Thanks, Phil and ProDomus Software for your generous support to the Clarion Developers Challenge and the Clarion Community. Bo will also be receiving a copy of

Product Scope 32 PRO, single user, spreadsheet license from David Troxell and Encourager Software.
Posted Thursday, November 20, 2003

LogFlash C6 Gold Compatible

LogFlash 2.5 has been released, and is now compatible with all versions of Clarion from CW20 to Clarion 6.
Posted Thursday, November 20, 2003

MessageBox 1.75

MessageBox 1.75 is a minor release allowing you to set default icons for the Clarion button equates, and a couple of very minor fixes.
Posted Thursday, November 20, 2003

xXPpopup v1.0

xXPpopup v1.0 is a class and code template to add Windows XP style popup menus to your application. No DLL, just Clarion source code. Supports single EXE, multi-DLL (local, standalone), 32 bit. For C5, C5.5, and C5 (ABC only). Demo available. Price is $59, available at ClarionShop.
Posted Thursday, November 20, 2003

Favorite Embeds Template

BFET, BigMyFavoriteEmbedsTamer replaces the BigEmbedsTamer template. Current users will receive free upgrade. $39 USD. Supports C5 through C6 ABC chain. This is a collection of embed buttons to go directly to your favorite embeds. Add the global template to your app, and the extension appears in each procedure with the embed buttons. You can leave it in, or remove it before release.
Posted Thursday, November 20, 2003

Holiday Treats From Gitano Software

With any purchase of $99 or more you will receive 25% discount on your purchase (deducted when we charge your card), a free PowerSearch license, and a $50 gift certificate for www.1stLogoDesign.com. This is a minimum value of $93.75
Posted Thursday, November 20, 2003

Gitano Software Utilities Update

The following utilities have been updated: gCal, C6 Gold Compatible. Local compile fixed C6 and C55; gCalc, C6 Gold Compatible. New local compile C6 and C55; gFileFind, C6 Gold Compatible; gQ, C6 Gold Compatible; gSec, C6 Gold Compatible; gReg, C6 Gold Compatible, select path for greglang.tps; gNotes, will be released in the next couple of days. All updates are free if you are within the one year upgrade period. All others can upgrade for a nominal fee.
Posted Thursday, November 20, 2003

Native XP Panel Source Code

The source-code for XP-Taskpanel is now available.
Posted Thursday, November 20, 2003

EasyExcel 3.03

EasyExcel 3.03 is now available. Changes include: Chart template bug fix - incorrect code generation when selected two-colour gradient other than Horizontal, Vertical or Mixed; Fixed export of HUGE (more than LONG) numbers into Excel; Fixed RunAutoMacros method error on some Excel versions;Fixed CopySheet method GPF; New methods include Calculation, ClearRange, SetPrintArea, SetPrintAreaA1, GetCellsProtection, CopyRange and Paste; Changed Page Setup template - added Print area tab. Compiled for C6 gold release.
Posted Thursday, November 20, 2003

CPCS v6.00 Gold Released

CPCS has released the Gold build of v6.00, and all associated add-on products, to coincide with the Gold release of Clarion 6. These files do not contain expiry dates. All CPCS C6 Beta testers can download and install these files using the same codes provided for previous beta builds. Licensed users of v5.57h or v5.16, who were not beta users can contact CPCS via email for install codes. Licensed users of versions earlier than v5.57h and v5.15 must upgrade ($100 US) to receive these versions.
Posted Saturday, November 15, 2003

Clarion6 Gold Pre-Release Patch

A C6 Gold pre-release patch is now available to EA program participants.
Posted Saturday, November 15, 2003

All CapeSoft Accessories Ready For Clarion 6 Gold

All the CapeSoft Accessories have been built for the new Clarion 6 Gold release. Please note the status of each product in terms of Clarion 6 thread-safeness. Possible options include: Built: install is made, example compiles, no other testing done; Tested: tested internally, seems 'OK'; Approved: no known problems; Released: for commercial use with Clarion 6 gold release.
Posted Saturday, November 15, 2003

BST Scheduler Resource Manager Suite

BST Scheduler and Resource Manager is officially released. Click on the BST Suite button on web site.
Posted Saturday, November 15, 2003

ImageEx 2 Anniversary Special

Today is the 1st anniversary of ImageEx 2, and to celebrate solid.software is offering ImageEx 2 for $159, which is 20% off the regular price. This special is in effect until November 30. Everyone who buys ImageEx2 by the 30th will also get a free license for SysTrack.
Posted Saturday, November 15, 2003

Fenix 1.3 Released

Fenix 1.3 is now available. Changes include: Improved support for TPS Files; Better SDK 1.1 ODBC Framework support; Localization improvements; Crystal Report Template improvements; SQL code template enhancements; Calendar and CSS issues addressed; New email features; BO-LAYER As Procedure, more embeds, some bug fixing with virtuals,

autoNumbering at BO; New SQLClass method ExecScalar(string); New Utils class method IsDate(); Added new Crystal class; Page template changes; Improved support for list control (mapped to datagrid); Automatic validators for controls with REQ attribute; Wizards now use FileDrops; Project Compilation like VS.Net; Better VS.NET Integration (project template as procedure template allows also Embeditor capabilities); HTTPS support; Browse template changes including better locators and smart column default properties; Form validations; Web services bug fix; IE tree control adds XML support for extended characters; more WebConfig options.
Posted Saturday, November 15, 2003

XP Taskpanel

XP Taskpanel is a native control for Clarion which emulates the task panels that are found in Microsoft's Windows XP. Taskpanels provide a flexible way of presenting a group of related tasks to your users with a graphically rich, modern looking, visually appealing way of presenting logically grouped sets of choices. This control operates under all (32-bit) version of Windows. Features include: Unlimited number of headers and tasks; Supports icons in headers and tasks; Blue and white styled headers; Expand/Contract headers; Easy to use control template; Multiple controls per window; Mimic buttons; Hide/Unhide headers and tasks; Get and Set header and task info at runtime; Automatic scrollbar; Mousewheel scrolling; Pure Clarion/API-code, no OCXs; Real (resizable) gradients; Customizable colors; Windows 95/98/NT/2000/XP. Demo available.
Posted Saturday, November 15, 2003

ProDomus Software Named Third-party Vendor of the Week

Phil Will and ProDomus Software have been named by the Clarion Developers Challenge Football Contest as the Third-party Vendor of the Week. Phil Will and ProDomus are awarding this week's winner of the Clarion Developers Challenge Football Contest a copy of The PD "Universal" Drop Edit Controls.
Posted Saturday, November 15, 2003

Clarion Developers Challenge Week 10 Winners

First place for Week 10 of the Clarion Developers Challenge goes to Dave Beggs. It was a tough week for all competitors, but Dave pulled out the win this week with 11 correct picks, while Alejandro Contreras was a close second with 10 correct picks. For finishing first in this week's Clarion Developers Challenge, Dave will be receiving a copy of Localizer from Roelf Du Preez and Pea Brain Software.
Posted Saturday, November 15, 2003

EasyMultiTag 2.04

EasyMultiTag 2.04 is now available. Changes include: New embed points for tagging processing; new methods for tagging processing (OnAction, OnInvert, OnFilter, OnTag, OnTagAll, OnUnTag, OnUnTagAll); Example for tagging processing; Bug fixes. New demo and documentation.
Posted Saturday, November 15, 2003

Professional Wizards Half Price

You can now get the Icetips Professional Wizards at half price. With the release of Clarion 6,

with its new wizards, things have changed and Icetips has decided to make some changes also. The Professional Wizards have been dropped by 50% in price, down to $199.00. The Professional Wizards are both application and procedure wizards that can cut down on your development time by standardizing the look and feel of your application. They can also add any kind of global and procedure extension templates for you while they create applications or procedures. The Standard Wizards have been removed from the Icetips online store and are no longer a commercial product. Instead they are now freeware. A C6 compatible version of the Standard Wizards should be available before Clarion 6 is distributed.
Posted Saturday, November 15, 2003

Easy3DStyle 1.05

Easy3DStyle 1.05 released is now available. Changes include: New 3D string/3D prompt controls; Names of two Easy3DStyle class methods changed (inVisible and Visible). This is a free upgrade for all registered customers.
Posted Saturday, November 15, 2003

ABC Free Templates Final Pre-Gold Beta

The ABC Free Templates version 6.00 beta is now available. Changes include: Added ICriticalSection logic to "ThreadLimitGlobal" template; Added AttachThreadToClarion calls to NT Service"template (this provides support for C5.5 and Clarion 5 application migration, but it is recommended that the new Capesoft commercial template be used instead); Added embed points to BrowseInvertCheckOnLeftClick template; Reworked MiscEntryFieldClipboardSupport and related vsClipboardClass to allow customization of popup and translation.
Posted Saturday, November 15, 2003

Flash Videos Demo C6 Features

The new SoftVelocity web site includes a number of Flash videos demonstrating new C6 features.
Posted Monday, November 10, 2003

Clarion/ASP 1.3 December Release

Clarion/ASP 1.3 is in the documentation phase and is scheduled for a December release.
Posted Monday, November 10, 2003

DevCon 2004

The SoftVelocity web site reports that DevCon 2004 is in the planning stages.
Posted Monday, November 10, 2003

xWindowSettings 2.11b

xWindowSettings 2.11b is now available. This release fixes a GPF after deleting a record.
Posted Monday, November 10, 2003

XPMenu Updated

Customers who purchased XPMenu should have received an email with update data. If you haven't received this, but you did purchase the XPMenu, please contact Ronald.
Posted Monday, November 10, 2003

Clarion Essentials Course for Cape Town Rescheduled

The next Essentials Course for Cape Town has been re-scheduled for Nov 24-27, 2003. The course will be held at FTI House, Kenilworth, and costs R4800.00. Two more bookings are needed to finalize.
Posted Monday, November 10, 2003

EasyResizeAndSplit 2.03

EasyResizeAndSplit 2.03 is now available. In this release, splits alignment strategies allow you to define split strategies for the controls which will applied - resize controls proportionally or move controls.
Posted Monday, November 10, 2003

xToolTip v1.5

xToolTip v1.5 is now available. This release fixes a bug where a tip contained CRLF pairs and no Title or icon was used. Updated demonstration program and install kits for Clarion 5, Clarion 5.5 and Clarion 6 are available.
Posted Monday, November 10, 2003

Gradient Released

The Princen Group ICT has released its Gradient product, which lets you apply color gradients to all frames, buttons, groups etc. You can create and exchange the gradient definitions from app to app using XML with import and export facilities. For demo/purchase select the Clarion button on the left side menu.
Posted Monday, November 10, 2003

Free Clarion 6 Screen Saver

Get a free Clarion 6 screen saver from Gitano Software.
Posted Monday, November 10, 2003

Pea Brain Software Named Third-party Vendor Of The Week

Roelf du Preez and Pea Brain Software have been named by the Clarion Developers Challenge Football Contest as the Third-party Vendor of the Week. Pea Brain is awarding this week's winner of the Clarion Developers Challenge Football Contest a copy of Pea Brain's Localizer.
Posted Monday, November 10, 2003

BMT Menu Template Released

Comsoft7 announces the release of its BMT MenuTree Control Template. This template takes all the items in your drop down menu and places them in a Clarion Tree listbox format similar to the folder view in Outlook Express. It is a control template that drops on a generic window, with all properties exposed so you have complete control over how it looks and feels. Promotion price of $39 until November 30, then $79.
Posted Monday, November 10, 2003

Replicate 1.25

Capesoft's Replicate version 1.25 is now available. Changes include: A fix to a spurious

compression error; Provision for more than 65535 log files; Fixed GPF caused when receiving a request file.
Posted Monday, November 10, 2003

New CapeSoft NT Service Product Expected Nov 17

CapeSoft is at work on a template that will let you start your application as a normal program, and as a service which can be started at boot up, without the need for the user to log on. Release is scheduled for 17 November 2003.
Posted Monday, November 10, 2003

RADrace 2003: Fenix Finishes Second

Last Friday and Saturday the Benelux RADrace took place at the Advanced Development Centre in Utrecht, The Netherlands. From the original 14 teams that applied, 10 actually started on Friday morning. Friday evening 5 teams were send home, but both RADventure Teams stayed. RADventure was participating with a Fenix Team (Sebastian Talamoni and Erik Pepping) and a Clarion team (Arie Rens and Peter Rakke). Fenix finished second, just after Edcubed. Unfortunately, standard Clarion ended fifth due to software problems with hard disk space in combination with Oracle database. Normally they probably would have finished much higher.
Posted Monday, November 10, 2003

Clarion Developers Challenge Week 9 Winner

First place for Week 9 of the Clarion Developers Challenge goes to Dean Burgess. Once again, the Clarion Developers Challenge wasn't decided until the final moments of the Monday night game. The prize award this week is cpTracker from Greg Bethume and Berthume Software. Dean has had my eye on this gem for awhile now and is much appreciative of the award. However, he'd like to re-donate the prize back into the Clarion Developers Challenge to be awarded to the player who participates each week in the Clarion Developers Challenge from here until the end of the season and finishes with the lowest total score. That's right, the rules are reversed for this special award. If you finish last, you win!
Posted Monday, November 10, 2003

Press Release Service

Sue Pichotta points out this press release service which will distribute a press release to over 1000 editors of software-related publications, or to editors in a variety of other markets. Lists can be fine tuned to include just the editors in your area of interest.
Posted Monday, November 10, 2003

Promotion Offer From Ingasoftplus And EC Software

From November 3, 2003 until December 31, 2003 you can buy Ingasoftplus's EasyHelper at 20% discount, and also get the same 20% discount on Help & Manual from EC Software
Posted Monday, November 03, 2003

Clarion Developers And IT Companies

Use this map-based page to locate Clarion programmers in your area.
Posted Monday, November 03, 2003

Clarion Jobs Page Updated

The Clarion Jobs Page has been updated with new listings.
Posted Monday, November 03, 2003

EasyResizeAndSplit 2.02

EasyResizeAndSplit 2.02 is now available. Changes include: Optimized dynamic resizing behavior; CR6 support.
Posted Monday, November 03, 2003

CopyFlash 2.3 Released

CopyFlash 2.3 is now available. CopyFlash is now compatible with all versions of Clarion from CW20 to Clarion 6. Price is $49 and there are no runtime royalties.
Posted Monday, November 03, 2003

EasyListPrint 1.07

EasyListPrint ver 1.07 is now available. Changes include: Support for Legal (8-1/2 in. x 14 in.) and Letter (8-1/2 in. x 11 in.) page size; MS Excel export engine rewritten to use COM interface (using TopSpeed C++); Fixed problem with the standard REPORT export in C6.
Posted Monday, November 03, 2003

EasyReport 1.02

EasyReport 1.02 is now available. New features include: Support for C6; Custom pattern strings replacement at run-time.
Posted Monday, November 03, 2003

Clarion News Group Archives (NNTP)

The old Clarion newsgroups have been restored. This data is being made available to all in a read-only format. Use this site to do full text searches.
Posted Monday, November 03, 2003

Clarion News Group Archives (NNTP)

The old Clarion newsgroups have been restored. This data is being made available to all in a read-only format.
Posted Monday, November 03, 2003

File Manager Beta 3.25

File Manager 3 version 3.25 beta is ready and available for download. Changes include: Template now automatically adds the FM2=>1 Project Define for C55 and above; Template now automatically runs the Support ABC Template Utility; Template adds support for changing file prefix; Fixed MySQL Drop Index syntax; Fixed MySQL initial gODBCFile creation; Added support for marking a field READONLY for AutoNumbered fields; New version of BDE available.
Posted Monday, November 03, 2003

xWindowSettings v2.11

xWindowSettings v2.11 is now available. This version includes CR6 support, and translation

via LNG file.
Posted Monday, November 03, 2003

CapeSoft Free Stuff Page

CapeSoft now has a page dedicated to free stuff. This page also includes links to the Tip of the Week and opinion articles.
Posted Monday, November 03, 2003

Object Writer Version 2

Object Writer is a free CapeSoft Accessory for Clarion Developers, which makes it easy to create and use reusable classes (objects) in your applications. Version 2 includes: A new template called Object User, which allows you to use your classes in your procedures or to create Global classes; The ability to modify this Object User template and ship it with your classes so that other's can add it to their application; Improved documentation: 17 pages of documentation, two Jump-Starts, one example and all the info your need to create and use classes in your application.
Posted Monday, November 03, 2003

Safe Reader & Safe Writer Version 2

Safe Reader (free) and Safe Writer ($39) have been revamped. These applications allow you to decrypt / encrypt Safe files. A Safe file is a password protected file that uses 168-bit DES encryption. This ensures maximum safety for your files and documents.
Posted Monday, November 03, 2003

Capesoft Replicate

CapeSoft Replicate is a Clarion 3rd Party Accessory that provides an automatic, driver independent, file-version independent, mechanism for replicating the data in two or more databases. This means you can run two or more databases concurrently with the users making changes to the tables, without having to worry about merging the tables together. Basically, Replicate logs your changes, adds and deletes and then using a transport manager of your choice (ftp, email or file copy), exports the changes to the other sites, where the changes, adds and deletes are imported to that data set. This all done completely automatically without your users having to do anything. You can easily configure Replicate to make a complete running backup of your data (a "mirror-site"). If your databases gets corrupt - re-apply the logfiles, and you're back up and running again. Replicate supports both offline and online environments. This means you don't have to be connected continually - your logmanager can use dial-up periodically to do the necessary logfile sending/receiving. Together with FileManager3 and Multi-Proj, you can use Replicate to replicate between databases with different filedrivers. This means that you can have a database with a SQL backend replicated to a TPS database.
Price: US$349
Posted Monday, November 03, 2003

CapeSoft Progress Goes Gold

CapeSoft Progress has gone gold. This replacement for the standard progress bars requires no hand coding. Features include: Real time previewing as you change the progress settings; Globally replace progress bars with a single style, or add custom progress bars to any window; Add as many types and styles of progress bars as you like; Built on the fast, robust CapeSoft

Draw engine. On special at $29 (until 15 November when it becomes $39). Progress requires that you have CapeSoft Draw. Draw & Progress Bundle is on special at $119 (until 15 November when it becomes $129).
Posted Monday, November 03, 2003

Taboga BarCode Library 1.1

Taboga Barcode Library version 1.1 has been released, and is now a commercial product. With release 1.1 the library is compatible with the Legacy template chain. A number of bug fixes have also been included, and the template is now compatible with CPCS.
Posted Monday, November 03, 2003

Sterling Data Products At CR-6 Level

CopyFlash and BackFlash are now at CR-6, with the other templates to follow soon. Download details for this upgrade will be sent to customers individually.
Posted Monday, November 03, 2003

Berthume Software Named Third-party Vendor Of The Week

Berthume Software has been named by the Clarion Developers Challenge Football Contest as the Third-party Vendor of the Week. Berthume Software is awarding this week's winner of the Clarion Developers Challenge Football Contest a copy of cpTracker. cpTracker will assist you in managing your contacts, prospects, customers, sales, projects and tasks to the last detail.
Posted Monday, November 03, 2003

Insight Graphing Beta 1.24

Insight Graphing Beta 1.24 includes a number of bug fixes plus major improvements to Y-Axis scaling and a re-invented pie-slice method.
Posted Monday, November 03, 2003

Clarion Developers Challenge Week 8 Winner

First place for Week 8 of the Clarion Developers Challenge goes to Sebastian Streiger. Sebastian followed up his initial week in the contest where he tied for second place with 10 correct picks with an even stronger 11 correct picks and this week's win. Once again, the contest went late into Monday night and this time Sebastian pulled out the victory when Miami defeated San Diego. For winning Week 8 of the Clarion Developers Challenge, Sebastian will be receiving a copy of gCalc from Jesus Moreno and Gitano Software. For the overall standings, David Proudfoot continues to lead the pack with a total score of 69.
Posted Monday, November 03, 2003

Clarion Magazine

Moving Applications to Oracle: RI And AutoNumbering Part 1

by Jon Waterhouse

Published 2003-11-21

When you move an application from flat ISAM files to a client-server architecture using a database like Oracle as a back end, it is not just a matter of setting up the Oracle tables and pointing your dictionary at them. One concern is making sure that what you want to be treated as atomic transactions are actually treated that way. Another is making sure that your relational integrity (RI) is enforced in the best way possible. This article will look at two aspects of migration that you will have to deal with in short order: relational integrity constraints and auto-numbering.

Relational integrity

The relational integrity constraints imposed by the ABC templates boil down to the following:

- When you delete a parent record: the delete is disallowed (`RESTRICT`) or child records are either deleted (`CASCADE`) or have their parent link fields set to `NULL` (`CLEAR`)
- When you change the primary key on a parent record (which is referenced by foreign keys on the child records); the change is disallowed (`RESTRICT`), changed in child records (and possibly child relatives too) (`CASCADE`) or set to `NULL` (`CLEAR`).

When working with a backend database, such as Oracle, rather than flat files, you have the option to have the database server enforce these restrictions, rather than have the Clarion application enforce them. When working with Oracle, I prefer to let the server enforce these restrictions. One major advantage is that if there are other applications, developed in either Clarion or some other language, that access the same data, the RI rules will be applied consistently by the database, regardless of the application actually changing the data.

When you set up a relationship between files in the Clarion dictionary, the drop-down list has the options above, plus the same options with (Server) added. When you choose the Server options for looking after RI, your Clarion application backs off. It tries to save the current

record that it has been asked to save, and if that works, it is done. Your program expects any other RI activity to be handled by the server. If you choose to enforce RI locally, then after successfully saving the current record the application will go on to try and make the required changes to all of the related child records. A failure in any one of these child files will lead to a rollback.

So, if you choose to enforce RI in Oracle, how do you go about it? There are two things necessary. The first is setting up a foreign key constraint on the child table in Oracle. For example, say you have a table `ORDER_LINE` and a table `ORDERS`, where the order_no field in order_line has to match `ORDER.ID`. You would implement the constraint as:

```
ALTER TABLE order_line
ADD CONSTRAINT order_fk
FOREIGN KEY (order_no)
REFERENCES order (id).
```

This will ensure that when someone enters an order line it has an `order_no` that can be found in the `ORDER` table. It will also prevent anyone deleting (or changing the key value of) rows in `ORDER` that are referenced in `ORDER_LINE`. This is the same as the "on delete restrict" setting in the Clarion dictionary. For the restriction on order_line to work consistently it is also necessary to put a `NOT NULL` constraint on the order_no column. If this constraint is not present, and you enter a `NULL` value, Oracle won't try to enforce the constraint, since it knows it cannot find a value that matches – in Oracle, `NULL` is not equal to `NULL`.

You can also add to the foreign key constraint one of the two options `ON DELETE CASCADE` or `ON DELETE SET NULL`. These take care of the two other possible options for RI on deletes (Cascade and Clear) in addition to the `RESTRICT` option for both updates and deletes.

Oracle, in its wisdom, does not provide a built-in method for cascading changes on update. The theory is that a primary key should not change, so there should never really be any need for cascading updates, unlike cascading deletes, which you might need when, say, you are archiving records. You can achieve cascading updates in Oracle, but it's a bit tricky. There are basically two methods, both of which involve triggers.

Triggers are SQL functions that can be set up to carry out one or more other activities when a certain event happens. In particular, they can be set up to do extra things before or after inserts, changes and deletes. They can also be used to carry out something instead of a particular action. For example, a before delete trigger could be set up to mark a record as deleted (using a column in the data) rather than actually deleting it. One of the things that is not possible in a trigger is a commit, or any Data Definition Language (DDL) statement that would cause a commit. Thus anything that is carried out by the trigger is logically part of the same transaction. A `CLEAR` constraint on update can be accomplished very easily using a trigger. Here's an example:

```
CREATE TRIGGER clear_on_id_change
BEFORE UPDATE OF id ON order
```

```
FOR EACH ROW
BEGIN
  Update order_line SET order_no=NULL where order_no=:old.id;
END;
```

This trigger will only fire when `ORDER.ID` is changed. For all the rows where it is changed, the child rows that reference that parent row will have their order_no column updated to `NULL`. Obviously for this to work the `order_no` column should not have the not null attribute.

Following on from the approach above, it seems like a similar trigger could be written to accomplish cascaded updates:

```
CREATE TRIGGER cascade_id_change
BEFORE UPDATE OF id ON order
FOR EACH ROW
BEGIN
  Update order_line SET order_no=:new.id where order_no=:old.id;
END;
```

Unfortunately, cascading updates are not quite that simple. The `:old` and `:new` refer to the values before and after in the row you are changing in the `ORDER` table. However, this trigger will only work if you do something else first. By default the Oracle database enforces constraints immediately. The trigger as written will fire before (it's a before update trigger) the initial change is made. However, under normal circumstances it will fail because it is not possible to change the children's link (foreign key) field to a parent ID that does not yet exist since this violates the foreign key constraint. The trigger would also fail if it was declared as an after update trigger, because as soon as the update changed the parent value, the children (in `order_line`) would become orphaned, causing a rollback.

The way to get around this is to make the foreign key constraint `DEFERRABLE`, and to set the constraint to `DEFERRED` for this transaction (`SET CONSTRAINT order_fk DEFERRED`). This tells Oracle not to enforce the constraint until the commit happens, by which time you expect to have everything sorted out. Deferred constraints have been available since Oracle 8 (Current version of Oracle is 9.2). Before then a trickier method had to be used.

Oracle wrote a little package to make this slightly tricky logic easy. The package uses a series of triggers and procedures so that the initial change (of the parent id) does not really happen. What happens instead is that a new row (with the new id) is inserted into the parent table. The child rows are all updated to become children of this new row, and then the old row (which no longer has any children) is deleted. This only works if you are updating the parent to a primary key value that does not yet exist in the table. In my experience about the only reason you would ever have to change a primary key is when you want to change it to a value that already exists (e.g. when one real customer has two data rows with slightly different names and you want to consolidate the two customers into the one they should have been). This is not something that can be accommodated by this package. However, changes of this sort can still be accomplished using the deferred constraint method (assuming you are working with Oracle 8 and up).

In summary, the equivalent in Oracle of a `RESTRICT` RI constraint is accomplished by setting up a foreign key using a `NOT NULL` column. The `Cascade` and `Clear` constraints for deletes can also be built into the foreign key constraint. Update constraints, if you really need them, require triggers.

Next week I'll continue this discussion with a look at autonumbering.

Jon Waterhouse has been using Clarion since the 2.1 days. His main work is as an economist, and he finds that Clarion is well-suited for applications which impose order on various sets of data. His projects include questionnaire data entry programs, classification software (assigning projects to groups), plus some more interesting scheduling applications. Jon has also used Clarion to link text information together, and is currently developing a program that will store linked snippets of WordPerfect documents and print custom documents composed of several of these snippets. He is currently working for the Newfoundland Government on a project to measure the performance of government employment programs.

Reader Comments

Add a comment

Clarion Magazine

The Windows API: Downloading Files Part 2

by Veronica Chapman

Published 2003-11-21

In Part 1 I described the Windows API functions that constituted a fairly robust File Download Manager, and I offered a working implementation of the mechanisms described.

In this part I will take the process further by describing how to handle the situation where the downloaded file has been compressed for transfer efficiency, and needs to be automatically expanded for normal use.

Overview

As I did in Part 1, I propose to start by reviewing the Windows API functions that are required. These are (in order of usage):

```
LZOpenFile()
GetExpandedName()
OpenFile()
LZCopy()
LZClose()
CloseHandle()
```

The LZ prefix identifies these Windows API functions as supporting the Lempel-Ziv Compression Algorithms, which are Microsoft's preferred method of compression. If a file name has an underscore as the last character of the file name extension, then (by Microsoft convention) it will almost certainly be an LZ-compressed version of the actual file. The actual file means the final, usable, expanded file (which will have a regular alpha character as the last character of *its* file name extension).

For example, MMPLAYER.HL_ will expand to MMPLAYER.HLP, which is the Help File for the Multimedia Player.

When a file is compressed, using Microsoft's COMPRESS.EXE utility, the original (i.e. the expanded) name can be embedded. The `GetExpandedName()` function will extract any original embedded name. This provides the means of automating the correct naming of the file when copy/installing it locally.

LZOpenFile()

The function of `LZOpenFile()` is to create, open, reopen, or delete a specified file. According to the documentation, it should support any kind of file, however I have found it best to limit its usage to only those that are LZ-compressed. The `OpenFile()` Windows API, and of course the regular Clarion `OPEN` will do the same job for all regular, uncompressed files.

The reasons why I would not use this API on regular file is that I have encountered situations where `LZClose()` and/or

CloseHandle() report the file is closed, and yet a Clarion OPEN fails with a "File already open" error. However, when limiting LZOpenFile() to compressed files, I have encountered no problems.

I should, however mention, that I always pair up LZOpenFile() with LZClose(), OpenFile() with CloseHandle(), and Clarion OPENs with Clarion CLOSEs.

The Prototype of LZOpenFile() is:

```
LZOpenFile(*ASCIIsz            szFileName,|
           *OpenFileStructure  ofStruct,|
           WORD                wStyle),hFile,|
                                  PASCAL,|
                                  RAW
```

szFileName is not complicated. It is the simply the file name plus extension (e.g. 'MyFile.ext'). However, it does need to be in the 16-Bit MS-DOS 8+3 format (i.e. including the embedded '~' if it is a long file name).

The second parameter, OpenFileStructure (discussed below), contains the means for specifying the appropriate file path.

ofStruct is actually a GROUP, corresponding to a specification known as an Open File Structure. Its usage is very common throughout Windows API functions that support file-based operations. There is a TYPEd declaration for it in my WINAPIX.CLW. The GROUP is declared thus:

```
OpenFileStructure       GROUP,TYPE
cBytes                     BYTE
cFixedDisk                 BYTE
nErrCode                   WORD
nFileDateTime1             WORD
nFileDateTime2             WORD
szPathName                 ASCIIsz(OFS_MAXPATHNAME)
                        END
```

If WINAPIX.CLW is INCLUDEd in your .APP (with the "Equates" SECTION in Global Equates, and the "Prototypes" SECTION in the Global MAP) Then to embed these sections you use the following code in the global data embed:

```
INCLUDE('WINAPIX.CLW','Equates')
```

And the following in the global map embed:

```
INCLUDE('WINAPIX.CLW','Prototypes')
```

With the equates section embedded you can code a data declaration this way:

```
SourceOpenFileStructure LIKE(OpenFileStructure)
```

You will then be able to refer to the members of your SourceOpenFileStructure in the form:

```
SourceOpenFileStructure.szPathName = 'folder\path\..\.. '
```

In the case of my example, a compressed download would have been stored (temporarily) in a binary file. This would be the source file of a copy-expansion process (the destination file being, of course, the expanded copy).

In order to run the expansion process, the source file needs to be presented to LZOpenFile(), and to do this an appropriate SourceOpenFileStructure must be prepared.

Inspecting the general form of this GROUP, above, you will see the following members:

cBytes - This member must be set equal to the size of the specific instance of the Open File Structure itself. The size will vary in accordance with the length of the last member, szPathName. In my TYPE declaration I used the maximum permissible size that Windows permits, namely OFS_MAXPATHNAME. OFS_MAXPATHNAME (Open File Structure Maximum Path Name) is a standard Windows API (EQUATE) definition. If you check in my WINAPIX.CLW you will find its actual value is 128 bytes. It isn't necessary to use all of these 128 available bytes to define the 16-Bit/MS-DOS version of the Folder Path. szPathName is a CSTRING, and therefore its *run-time size* (i.e. the string length in use) is always delimited by a binary zero terminator character. .

cFixedDisk - This member is interesting. Microsoft's definition is (and I quote all sources I have found): "[It] specifies whether the file is on a hard (fixed) disk. This member is nonzero if the file is on a hard disk.". Well, that's the best definition I found. A One is non-zero, so that's what I use.

nErrCode - This member "Specifies the MS-DOS error code if the OpenFile function failed". I use a value of One again. So LZOpenFile() would return 1 if the downloaded file fails to open.

nFileDateTime1, nFileDateTime2 - These members are only used to re-date a file. The standard dating mechanisms apply for the copy/expansion process, so I set these values to zero. (In older manuals these two members of the Open File Structure are often defined as "Reserved, do not use.")

szPathName - This member enables the Folder\Path\... to be specified. Because LZOpenFile() comes (historically) from the older Win16 APIs, it is necessary to set this value to the SHORTPATH() (i.e. the 16-Bit/MS-DOS) format. It is essential to ensure that, when setting this member from (say) the Clarion SHORTPATH() verb, you CLIP() the statement line.It is never a good idea to hand un-clipped strings (i.e. strings with trailing spaces) to any Windows API function. That's a fast track route to a hung application.

The next parameter to LZOpenFile is wStyle. wStyle is, quite frankly, badly named (remember I'm using the original notation as far as possible, so do not hold me responsible for these names). What it means, put simply, is how you want the file opened. It would be better named as wMode or something along those lines.

wStyle can take a number of values, which I quote below. Unusually (for Microsoft), their documentation is reasonably clear in this respect, and so I include their own definitions/descriptions of usage (Note they are all prefixed with OF_ to indicate that they refer to Opening a File- and note also that I have added some italicized comments of my own underneath):

OF_CANCEL - Ignored. In the Win32-based application programming interface (API), the OF_PROMPT style produces a dialog box containing a Cancel button.

OF_CREATE - Directs LZOpenFile to create a new file. If the file already exists, it is truncated to zero length.

OF_DELETE - Deletes the file.

OF_EXIST - Opens the file and then closes it to test for a file's existence. (This is actually quite useful, because the only way to do it in Clarion is – to the best of my knowledge – to define a DOS or ASCII File, and OPEN/CLOSE it in a Clarion "check for existence" function. This means always having a DOS File declared, usually in global data.)

OF_PARSE - Fills the OFSTRUCT structure but carries out no other action. (I'm not sure how. Only your APP knows the size of szPathName.)

OF_PROMPT - Displays a dialog box if the requested file does not exist. The dialog box informs the user that Windows cannot find the file, and it contains Retry and Cancel buttons. Choosing the Cancel button directs LZOpenFile to return a "file not found" error message.

OF_READ - Opens the file for reading only. (This is the value to use in the context of this article)

OF_READWRITE - Opens the file for reading and writing.

OF_REOPEN - Opens the file using information in the reopen buffer.

OF_SHARE_DENY_NONE - Opens the file without denying other processes read or write access to the file. LZOpenFile fails if the file has been opened in compatibility mode by any other process.

OF_SHARE_DENY_READ - Opens the file and denies other processes read access to the file. LZOpenFile fails if the file has been opened in compatibility mode or has been opened for read access by any other process.

OF_SHARE_DENY_WRITE - Opens the file and denies other processes write access to the file. LZOpenFile fails if the file has been opened in compatibility mode or has been opened for write access by any other process.

OF_SHARE_EXCLUSIVE - Opens the file in exclusive mode, denying other processes both read and write access to the file. LZOpenFile fails if the file has been opened in any other mode for read or write access, even by the current process.

OF_WRITE - Opens the file for writing only.

Putting this all together – in order to make practical use for most purposes - is actually fairly easy. In the current context, the data looks like this:;

```
SourcehFile             hFile
szSourceFileName        ASCIIsz('filename.ext')
SourceOpenFileStructure LIKE(OpenFileStructure)
dwMode                  WORD
```

The following statements initialize the structure (omitting an unnecessary variable declaration for wStyle):

```
szSourceFileName = CLIP('download file name and extension')
SourceOpenFileStructure.cBytes = SIZE(SourceOpenFileStructure),
SourceOpenFileStructure.cFixedDisk = 1,
SourceOpenFileStructure.nErrCode = 1, SourceOpenFileStructure.nFileDateTime1 = 0,
SourceOpenFileStructure.nFileDateTime2 = 0,
SourceOpenFileStructure.szPathName = CLIP(SHORTPATH('folder\path\..\.. '));
```

And here's the API call:

```
SourcehFile = LZOpenFile(szSourceFileName,|
                         SourceOpenFileStructure,|
                         OF_READ)
```

If LZOpenFile() succeeds, it returns a non-Zero, and *special* file handle, which here is stored in SourcehFile. This handle is special because it defines a compressed file, as opposed to a regular expanded file. LZCopy(), below, recognizes the special value handle, and turns on the expansion processes during the copy.

GetExpandedName()

The name of the destination file may very well be embedded in the compressed source file and, if that is the case, it should be extracted and used to name the destination file. In other words the name of the destination file may be – to some extent – tied to that of the compressed source file. This can be checked for, and often handled automatically, by means of the API function `GetExpandedName()`.

The mechanism does however, go back to the way the source file was originally created – namely how it was compressed in the first place.

You can compress files using Windows API calls, of course. However, serious automation for compressing files is better discussed along with setup installers and zipping utilities, etc. And, in any case, this article is about downloading files, not zipping up and uploading files.

If you do want to compress files (say to test the decompression code) you can do so easily using Microsoft's MS-DOS utility called COMPRESS.EXE. A copy of this utility is included in my example package. COMPRESS.EXE is easy to use. The command is:

```
COMPRESS <original expanded file name> /r
```

The "/r" parameter turns on auto-renaming, and hence the embedding of the "original expanded file name". To run COMPRESS.EXE on, say, a .HLP file (for example MyHelp.hlp), the MS-DOS command is:

```
COMPRESS MyHelp.hlp /r
```

In this case COMPRESS.EXE will create a compressed file name MyHelp.hl_, and embed the name "MyHelp.hlp" within it, in such a way as to enable `GetExpandedName()` to find and extract it.

The Prototype of `GetExpandedName()` is:

```
GetExpandedName(*ASCIIsz szSource,|
                *ASCIIsz szBuffer),sint,|
                                  PASCAL,|
                                  RAW,|
                                  NAME('GetExpandedNameA')
```

`szSource` specifies the name of the Source (compressed) file. I must confess that circumstances have never been such that I have had occasion to prepend a `PATH()` within `szSource`. Consequently I cannot state, for sure, whether or not the Folder Path can be included in `szSource`, and the Microsoft documentation (as ever) provides no clues. So be warned. You may need to change directories if the source file is not in the current directory.

`szBuffer` is where the embedded name will be placed – if it can be located. It will be in 16-Bit MS-DOS 8+3 format, and so the size of `szBuffer` needs to be 13 bytes (i.e. 8 + 1 (for the ".") + 3 + 1 (for the `CSTRING` zero-terminator))

Microsoft's documentation does *not* state that the source file must have been opened prior to calling `GetExpandedName()` and, since its prototype demands the file name as a string – rather than the opened file's handle – it is likely to operate on a closed file. However this is another aspect I have never tried – so take care.

Consequent to the above, here's a definition for `szBuffer`, taking `szSourceFileName` from the `LZOpenFile()` declaration (above):

```
szBuffer ASCIIsz('filename.ext')
```

Here's the API call:

```
IF GetExpandedName(szSourceFileName,|
                   szBuffer) = 1
... szBuffer contains the embedded File Name

ELSE

... there was no embedded File Name

END
```

If no embedded name can be found, `GetExpandedName()` does the utterly unhelpful thing of returning an uppercase copy of the source file name in its second parameter.

Its return condition is not particularly sophisticated. If it succeeds, it is documented to return a value of 1. If it fails, it is documented to return `LZ_ERROR_BADVALUE` (which, I can assure you, is different from 1).

OpenFile()

The function of `OpenFile()` is identical to that of `LZOpenFile()` for use on regular expanded files, e.g. the destination file.

The Prototypes of `LZOpenFile()` and `OpenFile()` are identical, and so is their usage in this context - except for the variances explained here:

a. The destination file's name might have been derived from the output of the `GetExpandedName()` function, above;
b. The destination file will require a `wStyle` (i.e. `wMode`) value equal to `OF_CREATE + OF_WRITE` (as opposed to `OF_READ`);
c. The file handle returned by `OpenFile()` is a normal one (as opposed to a special one), and will be recognised as such by the `LZCopy()` function, below.
d. I recommend that the file handle returned by `OpenFile()` should be closed by the API function `CloseHandle()` rather than `LZClose()`.

LZCopy()

The purpose of `LZCopy()` is to copy a source file to a destination file. If the `hFile` of the source indicates a compressed file, the this function creates a decompressed destination file. If the source file is not compressed, this function merely duplicates the original file (but, remember, this duplication can be into a different folder).

The Prototype of `LZCopy()` is:

```
LZCopy(hFile Source,|
            hFile Dest),LONG,|
                       PASCAL,|
                       RAW
```

... and since `LZOpenFile()` and `OpenFile()` will have provided the handles, the following code copies and expands the file:

```
IF LZCopy(SourcehFile,|
          DestinationhFile) < 0
```

If the copy failed, `LZCopy` reutrns an `LZERROR`, which is a value less than zero.

The various LZERRORs are

- LZERROR_BADINHANDLE: The handle identifying the source file is not valid. The file cannot be read.
- LZERROR_BADOUTHANDLE: The handle identifying the destination file is not valid. The file cannot be written.
- LZERROR_GLOBALLOC: The maximum number of open compressed files has been exceeded or local memory cannot be allocated.
- LZERROR_GLOBLOCK: The LZ file handle cannot be locked down.
- LZERROR_READ: The source file format is not valid.

LZClose()

This function closes the special file handle created by LZOpenFile() for the compressed source file. Its prototype is:

```
LZClose(hFile),PASCAL, RAW
```

In other words it is actually a procedure, having no return value. Therefore you can call it this way:

```
LZClose(SourcehFile)
```

CloseHandle()

This function closes the regular file handle created by OpenFile() for the expanded destination file. Its prototype is:

```
CloseHandle(hFile),BOOL, PASCAL, RAW
```

CloseHandle is a function, taking a non-zero return value to indicate success. However (as I said in Part 1), what does failure mean if the overall process has completely successfully? And even if it had not ... files still need to be closed. Consequently my use if it is to call it, and not give a second's thought to what it might return. Therefore I call it this way:

```
WBR = Closehandle(DestinationhFile)
```

Libraries

The downloadable source includes WEUSER32.LIB, WININET.LIB, RASASPI32.LIB, OLEACC.LIB, and LZ32.LIB. These make up the "shortfall" in the Clarion Linker's internal libraries (it will inevitably become out-of-date as new functions are created by Microsoft), and enable it to link to the Windows API functions prototyped in WINAPIX.CLW. Save them in a folder where your Clarion Linker can find them. (Not all of these .LIBs are used by this example, but it will do no harm to place them all together).

Summary of Part 2

As in Part 1, I will close with a summary of the foregoing as a working example excluding (for simplicity) all tests for error conditions. The same warnings apply as before – use this as a learning example, but if you want to create your own code, use the downloadable source as your starting point.

Globals:

```
WBR                         BOOL !(Windows Boolean Return)
DOSFileName                 STRING(300),STATIC
```

Locals & Code Statements:

```
szRemoteFile                ASCIIsz('filename.ext')
```

```
szBuffer                    ASCIIsz('filename.ext')
!
SourcehFile                 hFile
szSourceFileName            ASCIIsz('filename.ext')
SourceOpenFileStructure     LIKE(OpenFileStructure)
!
DestinationhFile            hFile
szDestinationFileName       ASCIIsz('filename.ext')
DestinationOpenFileStructure LIKE(OpenFileStructure)
!
! DOS (Temporary) File:
!
DOSFile           FILE,DRIVER('DOS'),NAME(DOSFileName),PRE(DOS),CREATE
Record               RECORD
BinaryData              GROUP
                           BYTE,DIM(BlockSize)
                        END
                     END
!
! Download Manager Window:
!
Window      WINDOW(' '),|
                  AT(,,378,38),|
                  FONT('MS Sans Serif',|
                  8,|
                  ,|
                  FONT:regular),|
                  CENTER,|
                  GRAY
               BUTTON('&Cancel'),|
                     AT(1,1,35,30),|
                     USE(?CancelButton)
               PROGRESS,|
                     USE(ProgressPercentage),|
                     AT(45,11,319,8),|
                     RANGE(0,|
                           100)
            END
  CODE
 OPEN(Window)
 Window{PROP:Text} = 'Expanding ...'
 DISPLAY()
 SETCURSOR(CURSOR:Wait)
!
 SourceOpenFileStructure.cBytes = SIZE(OpenFileStructure)
 SourceOpenFileStructure.cFixedDisk = 1
 SourceOpenFileStructure.nErrCode = 1
 SourceOpenFileStructure.nFileDateTime1 = 0
 SourceOpenFileStructure.nFileDateTime2 = 0
 SourceOpenFileStructure.szPathName|
                             = CLIP('Folder\Path\..\ etc.')
 szSourceFileName = CLIP(DOSFileName) ! ... from Part 1
!
 SourcehFile = LZOpenFile(szSourceFileName,|
```

```
                          SourceOpenFileStructure,|
                          OF_READ)
 !
 ! Open the - compressed - Source File:
 !
  IF SourcehFile > 0
     DestinationOpenFileStructure.cBytes|
                                 = SIZE(OpenFileStructure)
     DestinationOpenFileStructure.cFixedDisk = 1
     DestinationOpenFileStructure.nErrCode = 1
     DestinationOpenFileStructure.nFileDateTime1 = 0
     DestinationOpenFileStructure.nFileDateTime2 = 0
     DestinationOpenFileStructure.szPathName|
                           = CLIP('Folder\Path\..\etc.')
 !
 ! Look for embedded name and, if none, use a default:
 !
 ! (As noted in the foregoing text ... if no embedded
 !   name can be found, an UPPERCASE copy of the Source Name
 !   can appear in szDestinationFileName)
 !
     IF GetExpandedName(szSourceFileName,|
                        szDestinationFileName) NOT = 1|
      OR CLIP(szDestinationFileName)|
                     = UPPER(CLIP(szSourceFileName))|
        szDestinationFileName = 'default.ext'
     END
 !
 ! Create/Open the - expanded - Destination File:
 !
     DestinationhFile = OpenFile(szDestinationFileName,|
                              DestinationOpenFileStructure,|
                                 (OF_CREATE|
                                  + OF_WRITE))
     IF DestinationhFile > 0
 !
 ! Perform the expansion-copy:
 !
        WBR = LZCopy(SourcehFile,|
                     DestinationhFile)
        WBR = CloseHandle(DestinationhFile)
     END
     LZClose(SourcehFile)
  END
 !
  SETCURSOR()
  CLOSE(Window)
 !
  RETURN
```

Conclusion

In the third part of this article I will discuss how to cohesively merge Parts 1 and 2 into a working implementation if the entire process. I will also expand on some of the points that have been glossed over until now, in order to try to present a comprehensive explanation.

Download the source

Download WINAPIX.CLW

Download the source from www.merlinto.com

Download WINAPIX.CLW from www.merlinto.com

Veronica Chapman earned a B.Tech in Electronics & Electrical Engineering from Brunel University in 1968, and subsequently embarked on a career as a Programmer/Analyst, first writing code at machine level, and shortly thereafter working with real time systems and communications. By the mid '70s she was using languages such as COBOL and FORTRAN. Never a fan of BASIC or the C language, she discovered Clarion in the mid-90s and, ever since, has used it to create applications for the 16-bit (Win 3.x) and 32-bit Windows Platforms. An assembly code programmer from way back, Veronica discovered a cornucopia of very useful functions in the Windows API, and set about making these functions available to Clarion applications.

Reader Comments

Add a comment

Clarion Magazine

Reborn Free CLARION online

Moving Applications to Oracle: RI And AutoNumbering Part 2

by Jon Waterhouse

Published 2003-11-28

Last week I began a discussion of Oracle's Relational Integrity (RI) capabilities. This week I'll continue that discussion with a look at autonumbering.

There are two benefits of having the database enforce the RI constraints. The first is that your data is protected from some other application (or user armed with SQL*Plus) making changes that are inconsistent with relational integrity. Secondly, Oracle can carry out the required operations many times faster than a Clarion program can. For cascaded deletes, for example, a Clarion program will retrieve each of the child records one by one and then ask Oracle to delete each one. Oracle will get rid of all the unwanted records at the same time.

The downloadable example at the end of this article demonstrates the advantage of using Oracle constraints rather than relying on Clarion to enforce RI. The program first sets up a `PARENT` table by selecting 100 rows from the `ALL_OBJECTS` view. The table has a primary key of `OBJECT_ID`, which is numeric. It then creates a `CHILD` file by inserting 50 records for each key value in the `PARENT` table. The constraints on the tables are created by an `alter table` command. The program presents three methods of deleting all of the rows:

- A Clarion process using a Clarion "On delete cascade" constraint loops through the file deleting the parent records and their related children (the Oracle constraint is disabled)
- A Clarion process loops through the file deleting the parent records; the child records are deleted by the enabled Oracle foreign key constraint
- The records are deleted by a call to a single small SQL statement

Just to add a slightly more real-world dimension to the process the deletes take place based on the modulus of the `Object_ID`: first all IDs ending in zero are deleted, then those ending in 1, etc.

On my system the process using Oracle RI delete took about 8.26 seconds per 100 parent records (5000 child records). The single SQL statement with Oracle handling the RI took 5.7

seconds per 100 parent records. The Clarion method did not work at all. The child records related to the first parent ID were deleted fine, then Clarion got confused about which ID it was working with and failed with a "Record Not Found" error.

Auto-numbering

Creating primary keys using auto-numbering is something that Clarion programmers have become very comfortable with over the years. Many Clarion developers may not even realize that there are many SQL databases where primary keys are not auto-numbered arbitrary values, but consist of one or several entered fields. Partly, Clarion's autonumbering was a response to the quirkiness and poor performance experienced when dealing with multi-part keys in Clarion data files. In the SQL world auto-numbering to produce the primary key by which a record is linked to other data is frowned upon by relational purists, but tolerated. The theory is that each table is an entity, and there should be attributes of each instance of the entity that make it distinct. Auto-numbering is viewed as a bit of a cop-out. Still, most databases provide you with methods for autonumbering. In Microsoft's SQLServer there is an `IDENTITY` column attribute that handles auto-numbering. In Oracle auto-numbering is implemented using a *sequence.*

What happens if you use Clarion to autonumber the column you use as the primary key of a table? The ABC `PrimeAutoIncServer` method (which is called when you enter a form in insert mode, or when you choose Insert on a browse) does the following: queries the database to find the highest existing key value, increments it by one and inserts a record with the new ID number and all the other fields "blank" or with default values.

There are two major problems with carrying out this process in Oracle. The first is that "blank" records often cause problems. If you design your database properly many fields in your tables will not allow `NULL`s. For example, your order table should always require a customer; your order item should always have a price, etc. You *can* get around these constraints when you add your "blank" record by making up default values, but the price will be shoddy, error-prone data. You can save records with nonsense values for fields that should always have real values, but there is always the possibility that some of these crud-filled records will take up permanent residence in your table. The second problem with this technique is poor performance with a large number of users. If there are a lot of people inserting into a table, there are going to be a lot of requests to read one thing: the last value in the index. This is also an area of the disk that is going to be written to a lot: each time a new record is added there will be changes made specifically to that part of the index. Grabbing the next value from a sequence is, by comparison, very fast.

The Clarion method also has another built-in problem that will slow it down on busy systems. The insert is carried out in two separate operations: first the maximum value is retrieved from the table, then a record is added using the incremented maximum value. The possibility exists that another user can retrieve the same maximum value from the table *before* the first user gets around to inserting and committing its new record. This second user will then try to insert a record with the same key value as the first user has already added. This insert will be rejected. The `PrimeAutoIncServer` method gets around this problem by carrying out the whole process again if it fails the first time. By default it allows three tries. This is not just a

theoretical problem: it happens in practice. In the example application I have the number of attempts allowed set at 6: try reducing that to 1 and see how many records fail to get added. On my system it was close to three percent. Although this scheme manages to get records numbered and added, in the real world, if you have an application that responds to heavy usage by increasing the work the database is asked to do, you are asking for trouble.

Theory therefore says that using the Clarion method of auto-numbering will slow down your application rapidly as the number of users increases. The source that accompanies this article tests out this theory. Just for fun it also looks at what happens if you use Topspeed files for the same sort of load.

Before describing what the program does, I'll explain how you *should* do auto-numbering in Oracle.

The way to handle auto-numbering in Oracle is, as I said, to use a sequence. The sequence exists completely separate from your table. Its only function is to provide the Nextval in the sequence when it is asked. It may occasionally skip numbers, but it will never provide a duplicate value. When you want to insert a record you ask the sequence for the next value and put this value in the appropriate column. There are two ways of accomplishing this in Clarion. The first method is the one described in the Clarion help file. First your program asks for the `Nextval` in the sequence. When you save the record you use this value as the ID. This requires setting up a dummy table in Clarion with one field (a `ULONG`) to receive the value from the sequence. Your program gets the value passed to it using `PROP:SQL`, for example:

```
Dummy_table{PROP:SQL}='SELECT order_seq.nextval from dual;'
```

The second method is to set a trigger up in Oracle, for example:

```
CREATE OR REPLACE TRIGGER AUTONUM_ORDER
BEFORE INSERT ON ORDER FOR EACH ROW
BEGIN
  SELECT order_seq.nextval into :new.id from dual;
END;
```

If you have more esoteric auto-numbering needs (incrementing numbers for the second or third parts of keys), the Clarion method may start to look more attractive. However, the "add an empty record" Clarion method still has problems, as I noted earlier. The pure Oracle alternative would be to maintain a sequence for each of the numbering subsets. My guess is that any numbering scheme involving more than one field will cause more problems than it's worth.

Testing the theory

The theory suggests that the larger the number of contending users the worse the performance of the clarion scheme will be compared to the Oracle sequence scheme. The example program is set up to test this by taking a fixed number of records to add and looping through a series of "users". Each user gets to add their segment of the records. For example, if the number of

records to add is 1000, the program first spawns one "user" to add 1000 records, then 2 users to add 500 records each, then 3 to add 333 records each, and so on.

The actual inserts into the database are carried out by addrecords.exe. This is a little project-generated executable that is passed the number of records to insert, and the scheme for generating the id number. The records are added in a tight loop with a `YIELD` statement. The main program calls `RUN addrecords` for each user. A statement in a timer checks to see if all of the (1000) records have been added for that round of the loop. If so it goes on to the next round of the loop (with an increased number of users). To be more precise the check is to see if at least 97% of the records having been added: the number that are actually inserted can be less than the full amount both because of integer division (three users would add only 999 records) and because you *may* have some records not added because of the duplicate value problem (in the Clarion scheme). In pseudo-code what happens is this:

```
Tot_recs_to_add=1000
Num_users=1
:start
User_recs_to_add=Tot_recs_to_add/num_users
Loop k=1 to numusers
        Run addrecords(user_recs_to_add)
End !loop
Check 5 times a second
        If  at least 970 new records have been added
                Num_users=Num_users+1
If  Num_users > 10 then finish else go back to :start
        End if
```

The slightly complicated structure is required because while you want some of the actions to run concurrently (e.g. four versions of addrecords running to simulate four users), you don't want to start the five user scenario until the four user scenario has terminated.

Times can either be checked by calls to `clock()` at the beginning and end of the outer loop, or by looking at the times of the records added (one of the columns stored in the test table is the add time). I have used the first method.

On my machine, the practical results are as follows:

Using the Clarion auto-numbering: 1 user takes 5.69 seconds to add 1000 records

10 users take 9.34 seconds

Using an Oracle sequence to handle the numbering: 1 user takes 5.58 seconds

10 users take 8.33 seconds

The practical test does therefore validate the theory: while the sequence method is only 0.1

second faster for a single user, the advantage rises to 1.0 second when you reach ten users (which is a greater than 10% speed advantage). The tests were run on an HP clunker running NT4. Times are not that precise since the main timer only checks for completion five times per second.

Just for fun I set up a similar test to work with Topspeed files. If you need an example of how badly Topspeed files perform relative to a real database under multi-user access, this will do it for you. While using stream and flush is significantly faster than can be achieved using Oracle (about 1 second for 1 user), using `LOGOUT` and `COMMIT` around each transaction will kill performance when you add just the second user.

And not coincidentally, transactions are the subject of the next installment.

Download the source

Jon Waterhouse has been using Clarion since the 2.1 days. His main work is as an economist, and he finds that Clarion is well-suited for applications which impose order on various sets of data. His projects include questionnaire data entry programs, classification software (assigning projects to groups), plus some more interesting scheduling applications. Jon has also used Clarion to link text information together, and is currently developing a program that will store linked snippets of WordPerfect documents and print custom documents composed of several of these snippets. He is currently working for the Newfoundland Government on a project to measure the performance of government employment programs.

Reader Comments

Add a comment

Clarion Magazine

The Windows API: Downloading Files Part 3

by Veronica Chapman

Published 2003-11-28

In Parts 1 and 2 I described the Windows API functions that constituted a fairly robust File Download Manager, which can access and receive a chosen file, and also expand it automatically if it was compressed for transmission efficiency. I also gave minimal, but nevertheless working, implementations of these mechanisms.

In this final Part I will draw Parts 1 and 2 together, creating a more generalised implementation – and one that also includes a certain amount of error-checking. Where I tread ground already covered, I will provide just enough of a summary to keep the discussion moving along – you really will need to refer to those articles for the respective details.

Overview of the issues involved

In Parts 1 and 2 I did not dwell on the user interface – the window – at any great length. And, to be perfectly honest, there is not much to say about it. Nevertheless the following remarks are valid.

First, a window is not, in point of fact, *absolutely* necessary. A windowless implementation is perfectly possible – and actually valid – in certain circumstances. For example, where your application offers the facility of checking for updates, which might entail downloading (say) a very small file containing little more than latest version/date information, you wouldn't necessarily want to show the user any status information.. If an update is deemed necessary, and triggers an auto-download process from your application, then the resultant 'secondary' phase *would* need a progress window, for obvious reasons. But the first phase would be better implemented windowless, culminating in a message to tell the user that they were "already running the latest version", or that "there is a later version available ... do you want to download it now?"

So, having made those points, let me dismiss a windowless implementation, and dwell on it no further, by simply saying that the API function `FTPGetFile()` – used in place of `InternetReadFile()` – would be the better choice in such a case, because there would be little point in providing statistics. Furthermore `FTPGetFile()` would perform the task in the most efficient manner available, by downloading the entirety of the small notification file without interruption.

Now, returning to the windowed implementation as given in my example, it is, of course, very simple to construct – in Clarion.

The main point about it is that it should work *properly*. The Cancel button should cancel (requiring confirmation first, naturally). The Progress thermometer should start from the left-hand side and move, reasonably linearly, *fully* to the right, reaching 100% at the time the download is fully complete.The reasons for this are obvious. Downloads can be quite lengthy and the user should always be in the position of knowing exactly where they stand, and retain the ability to control the situation (without your application appearing to hang).

The Cancel methodology is straightforward. It can be driven from `CASE ACCEPTED()` in the standard way, and will work comfortably – provided that the `EVENT:Timer` and size of download segments are suitably chosen.

The suitable methodology for integrating the Internet access is therefore derived from the window's `EVENT:Timer`.

Other processes (e.g. the expansion process) will generally run too quickly for cancellation, or for progress notifications – even on the largest of files. (And furthermore, what is the point of the user spending 30 minutes downloading a massive file, but not being prepared to wait 10 seconds while it is expanded?)

And, last of all, the Window should be as informative as possible e.g. "File download in progress" is not as informative as "Connecting to <url> ...", and – subsequently - "Downloading <file>... nn% completed". The window caption is a perfectly good place for this information because (if the window is minimised) the user can move the mouse over the task bar icon and will be able to view an informative tooltip.

Although not coded in my example, *if* the start time was noted from `CLOCK()` at the beginning of the download process, then the elapsed time can be computed (and maintained on the display) each time a segment is processed (or, if you wish, once per fixed period). If the elapsed time is maintained, it can be used in a computation (together with the `TotalAmountDownloaded`) to compute the transfer rate.

During download, as the segments arrive, they must be stored. The suitable method for local storage is a DOS (i.e. binary) format local file, because this type of file will also support any format.

In my example, a temporary file is used, even if the download is uncompressed, i.e. downloaded in its final form. There are reasons for doing this. Firstly, if the remote file *is* compressed for transmission, then what is downloaded is (by definition) a temporary or intermediate file. It will need to be expanded, and the expansion process provides a built-in mechanism for final placement. Secondly, even if the file is *un*compressed the user may decide to cancel the operation. And it may be that an upgrade situation is in progress – in other words the user already has a previous (working) copy of the file, and believes that it will remain intact until a final commitment to overwrite is made.

Lastly, if you consider the detail of the standard Internet Explorer download process, you will note that you are offered the facility of saving to disk, and at that time you specify *where* to save. But the download actually stores into a temporary file and then *copies* to your placement choice before making Open, Open Folder and Close available. That is why you have a folder called Temporary Internet Files, and also why Windows API functions `GetTempPath()` and `GetTempFileName()` exist.

Consequently my example uses a temporary file into which the raw download is stored, and from which the final placement, or final placement plus expansion, is copied.

As mentioned above, there are two Windows API functions, `GetTempPath()` and `GetTempFileName()`, that could be called to:

1. Find the path to the Windows temp folder, and
2. Create a unique temporary file name.

I do not use these here simply because I do not see the need (in this instance). My basic argument is that it is simpler to place the raw download/temporary file in the current folder, and delete it once the final copy has been placed.

There is also a "choice of name" problem, which centers around the use of the API function `GetExpandedName()`. This function is useful to enable the expanded file name to be automatically extracted from a compressed file, but it has the characteristic that the temporary file name must be *exactly* the file name under which it is stored remotely – in order that `GetExpandedName()` works correctly. Furthermore the remotely-stored-compressed file must be downloaded into that *exact same name*.

If the remotely-stored file is uncompressed, things are a lot easier. Any reasonably unique name can be used. In this case I use 'fdlm.$$$'. If that name was ever likely to clash with my application (it would not, I can assure you), I would use some other temporary name. There is, of course, always the ".TMP" extension that could be used. ".TMP", by convention, indicates a temporary file.

The standard Clarion DOS-Binary driver is used to define this temporary file. Naming it is slightly convoluted because, as

explained above, the use of `GetExpandedName()` when necessary. However the benefits of using this function – in the form of being able to *fully automate* the expansion process in many cases – outweigh the convolutions involved.

When writing any application for a computer - particularly when the Internet is involved - many (and varied) error situations are possible. I have tried to make the example that accompanies this article as general as possible, and for this reason it encodes error situations and returns an error number. Based on this number it is possible to create any suitable user interface to support any "erroneous" situation.

The consequence of this is that you can, if you wish, copy the File Download Manager function and `MAP` prototype more or less as is, and paste it into any .APP of your choice as a source-coded function in its own right.

The example

I have already provided the majority of the File Download Manager's data declarations, together with explanations, in the text of Parts 1 and 2. In my example, the only significant addition is `ErrorNumber`, defined as a `BYTE`, and used to indicate any erroneous situation.

The example is a Clarion Project, contained in one Source Code File, and is fundamentally divided into five sections:

- a Global Section containing Data Definitions and MAP;
- a Main() Procedure which arranges to call the FileDownloadManager() function and to provide a simple example of supporting any error returned;
- the FileDownloadManager function itself, together with:
  - its ROUTINE named DownloadFile, and
  - its ROUTINE named ExpandFile.

The Global Section

This section arranges to `INCLUDE` my WINAPIX.CLW at the appropriate places – the "`Equates`" `SECTION` with the standard `EQUATE`s (EQUATES.CLW, etc), and the "`Prototypes`" `SECTION` within the global `MAP`.

It also inserts three extra `CURSOR` definitions (`:NoEntry`, `:Hand` and `:Busy`) that are not standard in my version of Clarion (CW2). They are included because, in my example, I use `CURSOR:Busy` – which is an hour-glass plus a pointer – to indicate "working ... but can be click-cancelled". (A little professionalism does no harm). However, if your particular Clarion includes these definitions, you can delete them.

As always, you will also find my "throwaway" `WBR` (Windows Boolean Return) variable. I never compile with warnings about not using `RETURN` values from functions. Although my method throws away many such `RETURN` values, the fact that I program `WBR` equal to them makes me *at least pause and consider*. Which is as it should be. Furthermore, while debugging, the throwaway value can be – temporarily – of use. Considerable amount of time is saved by having the throwaway value at my disposal (in `WBR`) during this period – especially when attempting to use Windows APIs for the first time.

Also in the global section you will find the DOS-Binary file name, because global (or module static) file names are demanded by the Clarion language.

Main() procedure

Within the Local Data section of this Procedure you will find variables which can be set – either by presets, or dynamically at runtime – to gain access to any file you are authorised to access. In order to make my example work, they are dynamically set to download a .PDF from `ftp://ora.com` (using your own e-mail address, which you need to set before any re-compilation). *The .EXE in the package that accompanies this article uses one of mine, but obviously I would rather you used your own.*

You can change the dynamic presets to suit your own experiments.

In this procedure there is a small prelude that simply arranges to bring up the WinHelp-style Help File accompanying the example. Most of what is said in that Help File has been said in this article.

Following that, the `FileDownloadManager` function itself is called, employing the call parameters that have been set or preset. A discussion of these is given below.

Upon return from `FileDownloadManager(),` this procedure decodes any `ErrorNumber` returned, and provides some simplified messages based on it. The point to note, is that `FileDownloadManager()` makes a point of returning the actual (i.e. true) name of the destination file in all cases, and this can be used for informative error messaging.

Having dealt with any erroneous situation, `Main()` closes down the example.

FileDownloadManager()

The `FileDownloadManager` function, which uses the `DownloadFile` and `ExpandFile` routines I discussed in the previous installments in this series, is coded as follows:

```
FileDownloadManager           FUNCTION (FnUserName,|
                                        FnPassword,|
                                        FnAgent,|
                                        FnServerName,|
                                        FnServerFolderPath,|
                                        FnRemoteFileName,|
                                        FnExpandedFolderPath,|
                                        FnDefaultExpandedFileName,|
                                        FnSourceCompressed,|
                                       *FnActualExpandedFileName)
! File Download Manager: Local Data:
!
BlockSize                     EQUATE(1024)
!
ErrorNumber                   BYTE
!
hInternetOpen                 hInternet
hInternetConnect              hInternet
FTPhFile                      hFile
!
szProxyName                   ASCIIsz(' ')
szProxyBypass                 ASCIIsz(' ')
!
szAgent                       ASCIIsz(101)
szServerName                  ASCIIsz(301)
szUserName                    ASCIIsz(31)
szPassword                    ASCIIsz(31)
szRemoteFile                  ASCIIsz(301)
!
dwContext                     LONG(0)
!
szBuffer                      ASCIIsz(BlockSize + 1)
FileBinaryData                GROUP,OVER(szBuffer)
                                 BYTE,DIM(BlockSize)
                              END
!
```

```
  ProgressPercentage            LONG
  !
  NumberOfBytesToReadHigh       LONG
  NumberOfBytesToReadLow        LONG
  FileSizeBytes                 SREAL
  !
  AmountDownloaded              LONG
  TotalAmountDownloaded         SREAL
  !
  szSourceFileName              ASCIIsz('filename.ext')
  SourceOpenFileStructure       LIKE(OpenFileStructure)
  SourcehFile                   hFile
  !
  szDestinationFileName         LIKE(szSourceFileName)
  DestinationOpenFileStructure  LIKE(OpenFileStructure)
  DestinationhFile              hFile
  !
  ! DOS (Temporary) File:
  !
  DOSFile            FILE,DRIVER('DOS'),NAME(DOSFileName),PRE(DOS),CREATE
  Record                RECORD
  BinaryData               GROUP
                              BYTE,DIM(BlockSize)
                           END
                        END
                     END
  !
  ! Dialogue Window:
  !
  Window  WINDOW(' '),AT(,,378,38),FONT('MS Sans Serif',8,,FONT:regular),CENTER,GRAY
           BUTTON('&Cancel'),AT(1,1,35,30),USE(?CancelButton)
          PROGRESS,USE(ProgressPercentage),AT(45,11,319,8),RANGE(0,100)
         END
   CODE
  !
  ! 'Safety Trap' in case parameters have not been set:
  !
   IF JustOfferHelp = True
      OPEN(Window)
      WBR = WinHelp(Window{PROP:Handle},|
                    szHelp,|
                    HELP_CONTENTS,|
                    0)
      CLOSE(Window)
      RETURN(0)
   END
  !
  ! Normal Operation:
  !
   szProxyName = ''
   szProxyBypass = ''
  !
   ErrorNumber = 0
  !
   IF FnSourceCompressed = True
      ! If so create Source of the same name
```

```
      szSourceFileName = CLIP(FnRemoteFileName)
   ELSE
      ! Use a fixed Source file name ...
      szSourceFileName = 'fdlm.$$$'
   END

  ! Use current Folder and the chosen Name
   DOSFileName = CLIP(PATH())|
               & '\'|
               & szSourceFileName
   CREATE(DOSFile)
   ! Open Source for WRITE, as a Temporary file,
   ! destroying any previous one
   OPEN(DOSFile,1h)
   IF ErrorCode()
      ErrorNumber = 1
   ELSE
      OPEN(Window)
      DISPLAY()
      DO DownloadFile
          ! Close the Temporary file ... it now
          ! becomes the Source for copying
      CLOSE(DOSFile)
      IF ErrorNumber = 0
  !
  ! Here we either straight-copy or expand-copy the
  ! Source File into its final placement (in other
  ! words install it):
  !
         IF FnSourceCompressed = False
            IF CLIP(FnDefaultExpandedFileName) NOT = ''
               FnActualExpandedFileName = FnDefaultExpandedFileName
            ELSE
               FnActualExpandedFileName = FnRemoteFileName
            END
            COPY(DOSFile,|
                 CLIP(FnExpandedFolderPath)|
               & '\'|
               & CLIP(FnActualExpandedFileName))
         ELSE
            DO ExpandFile
         END
      END
          ! Tidy up by removing the Source/Temporary file
      REMOVE(DOSFile)
      CLOSE(Window)
   END
  !
   RETURN(ErrorNumber)
```

The call parameters are defined below in the order stated in the procedure prototype

`(STRING) FnUserName` - this must be set to authenticate Web Site access.

`(STRING) FnPassword` - this must be set to authenticate Web Site access.

(STRING) FnAgent - this defines your application's identification within the Internet session i.e. your application's title.

(STRING) FnServerName - this defines the Server. It is the Web Site's base URL, e.g. "ora.com". Do not include the "www.", because "ftp." will be inserted automatically.

(STRING) FnServerFolderPath - this defines the file path of the remote storage location.

(STRING) FnRemoteFileName - this defines the name of the file as stored on the remote Server.

(STRING) FnExpandedFolderPath - this defines the file path for local storage at final placement.

(STRING) FnDefaultExpandedFileName - this must be supplied as a default in the case that a compressed remote file does not have its expanded name embedded. If FnSourceCompressed (below) = False, then this parameter is not essential but, if supplied, will be taken as an overriding name for final placement.

(BYTE) FnSourceCompressed - set True if the remote file is compressed, set False if not.

(*STRING - returned) FnActualExpandedFileName - on return, this will contain the true local name of the destination file.

(BYTE) Return Value = Error Number - value 0 = OK, or 1 to 10 possible erroneous situations as follows:

1 = The temporary/source file could not be created (use ErrorCode())

2 = Rejected by InternetOpen()

3 = Clarion APPEND error (use ErrorCode())

4 = Confirmed manual Cancel during download (not really an error)

5 = Rejected by FTPOpenFile()

6 = Rejected by InternetConnect()

7 = Rejected by LZOpenFile on the source (compressed) file

8 = Rejected by OpenFile() on the destination (expanded) file

9 = Rejected by LZCopy()

10 = "Clarion Close" error on destination file (use ErrorCode())

Where "use ErrorCode()" is *not* stated, you would need to consult the Microsoft Documentation to determine how to obtain more information. Most of it will come from the API call GetLastError(), but there are cases where other API calls provide more information.

In operation this function prepares the appropriate name for the source (or temporary) file, depending on whether or not the remote file is expected compressed.

The DOS-Binary storage file for the raw download is then CREATEd and OPENed for writing. The Window is OPENed and the DownloadFile routine is called.

On an error-free return from `DownloadFile` routine, the download file itself is Clarion-`CLOSE`d, and a branch is taken depending on whether or not expansion is required.

If no expansion is required, then Clarion's `COPY` is used to move to final placement. The name under which the raw download is finally placed depends on whether or not `FnDefaultExpandedFileName` was supplied. If it was, then that will be the name used, otherwise the final copy will be given the same name as the download itself. `FnExpandedFolderPath` is taken to determine the navigation for final placement.

Where expansion *is* required, ExpandFile is called.

In all cases the raw download is Clarion-`REMOVE`d from disk, the Window `CLOSE`d, and the current ErrorNumber is `RETURN`ed to the caller – in this case `Main()`, above.

DownloadFile

The `DownloadFile` routine retrieves the specified file using `InternetReadFile` in a timer loop. For a full discussion of this routine see Part 1.

ExpandFile

The `ExpandFile` routine uses the Windows Lempel-Ziv compression functions to expand a compressed downloaded file. These functions include `LZOpenFile`, `LZCopy`, and `LZClose`. For a full discussion of this routine see Part 2.

Conclusion

Using most of the above concepts it would be possible to create a File *Upload* Manager, just by turning a few things around. You would use `GENERIC_WRITE` in place of `GENERIC_READ` as the parameter to `FTPOpenFile()`, and `InternetWriteFile()` in place of `InternetReadFile()`. In this case the file size would be obtained by opening the source file and setting `TotalNumberOfBytesToWrite = BYTES(DOSFile))`.

I hope that you will now be in the position of being able to investigate the entire project in total, and eager to start some Internet-based access for your own applications.

Resources

A Microsoft Win32 API Bible is available from my web site. It is zipped up but, be warned, it is 5.6Mb in size. (It is, however, *well worth* the download time). This Bible is to some extent outdated, but it contains all the basic APIs, and is fairly well organized.

For Internet Explorer/Access information, my source is an HTML Help File called "inet.chm", with its index ‘inet.chi" that comes with the Microsoft SDK. My copy is over 32Mb in size. You can download it from debian.goldweb.com.au but the inet.chm there is quoted at 119Mb. inet.chm/chi provides you with a local copy of the exact same information offline as posted online in the MSDN Libraries.

Download the source

Download WINAPIX.CLW

Download the source from www.merlinto.com

Download WINAPIX.CLW from www.merlinto.com

Veronica Chapman earned a B.Tech in Electronics & Electrical Engineering from Brunel University in 1968, and subsequently embarked on a career as a Programmer/Analyst, first writing code at machine level, and shortly thereafter working with real time systems and communications. By the mid '70s she was using languages such as COBOL and FORTRAN. Never a fan of BASIC or the C language, she discovered Clarion in the mid-90s and, ever since, has used it to create applications for the 16-bit (Win 3.x) and 32-bit Windows Platforms. An assembly code programmer from way back, Veronica discovered a cornucopia of very useful functions in the Windows API, and set about making these functions available to Clarion applications.

Reader Comments

Add a comment