APOLLO™

SuccessWare Database Engine™ 2.10 for Delphi™

(**DEMO** Version 2.10.00, April 15th, 1996)

Overview:

Thank you for evaluating **APOLLO 2.10**, SuccessWare's award-winning Replaceable Database Engine for Borland Delphi. APOLLO was chosen as the **Best Delphi Add-In** in Delphi Informant's 1996 Reader's Choice Awards and is the choice of professional Delphi developers requiring small and fast support for FoxPro, Clipper, and/or HiPer-SIx data/memo/index files in their applications.

NOTE: This is a 16-bit demo edition of APOLLO 2.10 and is for use with Delphi 1.x (16-bit) only.

Applications created with this DEMO version of APOLLO 2.10 will only execute when Delphi is running. Except for this one limitation, this DEMO version of APOLLO 2.10 is completely functional and unrestricted.

APOLLO 2.10 allows Delphi database application developers to add high-speed access to FoxPro, Clipper, and SuccessWare's own HiPer-SIx file formats.

NOTE: *If you do not specifically require FoxPro or Clipper file format compatibility, you should consider using the HiPer-SIx file formats. The index (.NSX) and memo (.SMT) file formats used by HiPer-SIx are the smallest and fastest available in APOLLO.*

APOLLO 2.10 is a Delphi-specific interface for the SuccessWare Database Engine 2.10 (SDE2). Connections to SDE2, for most basic DBMS operations, are made through Delphi's own TTable component.

In fact, APOLLO 2.10's TApollo component is not required if you are only using the basic functionality provided through TTable. TApollo is only required if you will be using the extended functionality provided by SDE2 or any other APOLLO 2.10 features not supported through TTable itself.

The SDE used by APOLLO 2.10 is very small (less than 300k, for a typical application), yet completely self-contained. A Delphi application using APOLLO 2.10 does not require the Borland Database Engine (BDE) at all. However, if your application requires dBASE IV and/or Paradox file support as well, you can have both APOLLO 2.10 and the BDE in the same application.

Environmental settings for APOLLO 2.10 are handled through the TApolloEnv component. These include SetCentury, SetDeleted, SetSoftSeek, and more.

Installation:

To install APOLLO 2.10, run the included SETUP.EXE program from Windows. It will prompt you for an installation directory and will copy the new files into that directory.

The installation program will not overwrite any of your existing Delphi files. However, it may contain newer versions of the SuccessWare Database Engine .DLL files (APOLLO 2.10 uses the same engine .DLLs as the SDE 2.10). If so, the old versions of SDE 2.10's .DLLs currently residing on your system will be copied into a \BACKUP subdirectory.

Getting Started

Before using APOLLO 2.10, you must first rebuild Delphi's default Visual Component Library (VCL), COMPLIB.DCL, to add support for the SuccessWare Database Engine 2.10 (SDE2) used by APOLLO. This new VCL can be rebuilt to Use APOLLO With the BDE, or to Use APOLLO Without the BDE.

The SETUP.EXE program can optionally prepare Delphi's IDE to include the new VCL name and search path in order to use APOLLO 2.10. All you will have to do after exiting SETUP is to start Delphi and add the APOLLO.DCU file from Delphi's **Options | Install Components** menu to be up and running. If you experience any problems at all during installation of the .DCU file, follow the installation instructions as documented in the APOLLO 2.10 User's Guide and/or Help file (APOLLO2.HLP). Make sure to follow these instructions exactly and carefully.

After rebuilding Delphi's VCL, the APOLLO 2.10 Tutorial is a good place to start becoming familiar with APOLLO 2.10.

For those developers who have an Xbase background, the TApollo and TApolloEnv method names will seem very familiar. Those without an Xbase background may want to take a moment to scan through each of these methods to become familiar with all that APOLLO 2.10 has to offer.

Important Info for APOLLO 1.x Users!

This version of APOLLO uses different engine .DLL names from the previous release of APOLLO. See the online or printed documentation for details on these new files to be distributed with your APOLLO applications.

In general, your existing Delphi code using APOLLO 1.x will not need to change, with the following exceptions:

The second to last parameter of the TApollo.Index and TApollo.IndexTag methods has been changed from a Bool type (formerly the Unique flag) to an enumerated integer value (one of three values). The defined constants available for use in this parameter are now (IDX_NONE, IDX_UNIQUE, or IDX_EMPTY). This is in support of the new Roll-Your-Own (RYO) index features of APOLLO 2.10.

If your existing code was using **False** for this parameter, you will need to change it to **IDX_NONE**. If your code was using **True**, you would need to change it to **IDX_UNIQUE**. Also keep in mind that to use these defined constants, as with all other defined constants in APOLLO 1.x and 2.10, you must also include **DbiProcs** to the **uses** line at the top of your unit source code.

INTERNATIONAL USERS!

See the "Notes For International Users" section in the APOLLO2.HLP file for important information on using international sorting sequences in indexes when migrating from DOS-based applications.

New Query Optimizations in v2.10.00:

The Query Engine will use indexes to optimize queries in the following order:

1. The Current Index
2. An index with a matching expression, but no condition.
3. An index with a condition that exactly matches the query expression.

This allows v2.10.00 to always be accurate. You tell the system to also use conditional indexes that match the expression by setting:

```
Apollo1.SysProp( SDE_SP_SETUSECONDITIONAL, Pointer(1) );
```

(The default under v2.10.00 is 0).

This allows APOLLO to use conditional indexes where it normally would not. Be cautioned that the result set may differ from a SetFilter with the same expression. If all queries are supplied programmatically, this method allows for the most amount of optimizations. If there are some user supplied Ad-Hoc queries, it is best to not use this feature. By leaving the SysProp value to its default you ensure that all queries report the same results as an equivalent filter.

Optimistic Buffering:

APOLLO 2.10.00 has been enhanced to be more aggressive with its buffering schemes. By default, APOLLO will now optimistically buffer all tables opened in **EXCLUSIVE** and all tables opened **SHARED** on a **LOCAL** hard drive. When tables are opened SHARED on a local hard drive, the access speeds approach EXCLUSIVE times.

The basic philosophy behind optimistic buffering is that all file reads and writes are cached as long as possible. This allows APOLLO to visit a memory buffer for information instead of reading from or writing to disk. Besides speed benefits there are other subtle benefits. A file opened in shared mode on a local disk will only be opened once per task, no matter how many workareas are attached to the table. This saves on file handles and other system resources.

The current implementation of Optimistic Buffering uses a maximum of 30k per file for buffering. In future releases of APOLLO, this will probably be a selectable amount, but currently this is a fixed buffer size.

Caveats for Optimistic Buffering:

There are a few definable caveats for the use of Optimistic Buffering. These are the same caveats that inherent in the **Borland Database Engine (BDE)**.

1. APOLLO 2.10.00 does not detect if a **LOCAL** hard drive is a shared volume under a peer to peer network, such as Windows for Workgroups or Windows 95. If your application is opening a local table that is also to be opened by users logging into your local machine, turn Optimistic Buffering off (See : **New TApollo.SysProp Values**)
2. If data tables opened on a **LOCAL** hard drive are opened by APOLLO and another DBMS (i.e. Clipper or FoxPro), Optimistic Buffering should be off (See: **New TApollo.SysProp Values**). When APOLLO 2.10.00 opens a table with Optimistic Buffering on, file writes are cached and the physical files on the hard drive are not updated until the memory buffers are flushed. The net result is that a file opened under Optimistic Buffering is only guaranteed to be current for a APOLLO-based application.

New TApollo.SysProp Values:

For those of you unfamiliar with **SysProp** here is a basic outline. The **TApollo.SysProp** method provides a generic way to pass values into and receive values from the APOLLO subsystem. This flexibility allows us to add functionality to APOLLO without adding new functions.

For instance, there currently exists a SysProp manifest constant to retrieve the value of the SoftSeek flag. Without SysProp, to provide the same functionality, we would be required to add a new function to the DLL, such as sx_GetSoftSeekFlag, and a corresponding new GetSoftSeekFlag property to TApollo.

To use a SysProp value you must include **`DbiProcs`** to the **`uses`** line of your unit.

There are two important new values:
```
SDE_SP_GETOBUFFER  {Get the optimistic buffer flag}
SDE_SP_SETOBUFFER  {Set the optimistic buffer flag}
```

To turn Optimistic Buffering **OFF**:
```
TApollo.SysProp( SDE_SP_SETOBUFFER, Pointer(0) );
```

To turn Optimistic Buffering **ON**: (Default Setting)
```
TApollo.SysProp( SDE_SP_SETOBUFFER, Pointer(1) );
```

Improvements to the Query System:

The query system has been expanded in this release to include the optimization of QKeyVal() and the contains operator ($). Please refer to the APOLLO documentation for examples on these two operations.

Using the APOLLO2.HLP File:

After installing APOLLO 2.10, see the APOLLO 2.10 Help file (APOLLO2.HLP) for further information on configuring APOLLO 2.10 within Delphi's IDE. This same information is also contained in the printed APOLLO 2.10 User's Manual.

Sections within the Help file can be printed out by selecting **File | Print Topic** from the WinHelp menu.

The APOLLO2.HLP file contains a feature to allow users to look up any word or string at all that is contained within it. To use this feature, after using the Windows Help engine (WINHELP.EXE) to load APOLLO2.HLP, simply select the **Find+** menu option or speed button.

Getting Updates

SuccessWare posts periodic (and frequent) updates for all of its software products on both our in-house BBS (**909-694-6891, 14.4kbps, N-8-1**) as well as on our CompuServe section (**GO SWARE, LIB 3**). These updates are generally posted in the form of small patch files which will update your existing files.

To insure that you are always using the latest revision of any SuccessWare product you own, you should log into our BBS or our CIS section at least once a month.

In fact, if you experience any problems with a SuccessWare product, the first place to look is on our BBS or CIS section. If you see an update posted that is newer than the files you are currently using, you should download the update and see if it resolves the problem.

If all else fails, please contact SuccessWare for assistance.

Ordering Information:

Standard Edition:	US$179, plus FedEx shipping charges (US$20 within the U.S. and US$35 Canada).
Pro Edition:	US$269, plus FedEx shipping charges (US$20 within the U.S. and US$35 Canada).

(The Pro Edition includes the Delphi source code for the APOLLO 2.10 VCL components.)

We accept VISA, M/C, or AMEX.

Contact Information:

SuccessWare International
27349 Jefferson Avenue, Suite 110
Temecula, CA 92590

Voice:	800-683-1657 (or 909-699-9657 outside the U.S. and Canada)
Fax:	909-695-5679
BBS:	909-694-6891
CIS:	Team SuccessWare [74774,2240] or GO SWARE, Section #3
Internet:	TechSup@GoSware.com

Don't take our word for it. Here's what the users themselves are saying about APOLLO:

****

NOTE: These are actual unsolicited quotes from actual Apollo owners as posted publicly by them on CompuServe's DELPHI forum or our CompuServe section (GO SWARE, Section #3).

****

"When I tested (Apollo) against ODBC using 15,000 approx records, ODBC took around 5 minutes to move the record pointer to the last record, whereas using the Apollo driver it took around 1-2 seconds. More like the FoxPro I know." – Gregg Martin

"I would definitely recommend SuccessWare's Apollo product if you want to use either FoxPro CDXs or Comix CDXs if you're using a Clipper product. We currently are using the Apollo product at UPS as a replacement database engine to the BDE and it works great."
– James D. Sweatman

"Another success! When I first started developing a software package I realized that the BDE would be impractical. I needed to send it out on 1 disk, not 2-3 disks as the BDE would have required. I spent some time looking for db routines, but with no success. Apollo came to the rescue. And not just routines, but a tool that assimilates into Delphi and it's data-aware controls - now we're really moving! Apollo opens up a whole new arena of software development. Version 2.0 is very stable and works fast." – Dominic Lee-Delisle

"I am a developer that has been working with FoxPro for years. When Delphi came out it was obvious that it was time to change. Apollo is a great product because I can use the FoxPro files and distribute applications." – David Griffis

"I've had and used versions of CodeBase for years. However, I currently ONLY use Apollo with Delphi. The products match up much better and SuccessWare's support is MUCH better"
– Vic Williams

"I will not go into all the products I have tried, but I can honestly say my purchases are in the thousands of dollars for trying to find a solution for this legacy app and my customers. Recently I showed my Delphi-Apollo app to a consultant whose jaw dropped when she found out the price of Delphi and Apollo. She couldn't believe how

inexpensive these two tools were—and that they actually worked and are incredibly powerful." – Greg Hamilton

"I recommend getting the Apollo dbf engine if you want to continue working with dbf's. It makes the jump to Delphi easier, and it's smaller than the Borland engine." – Clayton Jones

"The deployment of one of the apps that took 3Megs+ with FoxPro is now less than a Meg with Delphi and Apollo and it's 3 times as slick." – Dave Bhatia

"I miss Access, but so far, Apollo/Delphi/Xbase provides all the capability I need, a very light footprint, and fast fast fast" – David McDonald

"Cool…. I'm a rookie at both Delphi and Apollo and I am up and running in 30 min."
– Tim Thompson

"Best Apollo yet! I've now got four multi-user Apollo apps running on our (Netware) 3.,11 network. One is coexisting nicely with a Clipper app; another is coexisting nicely with a FoxPro app. The two other apps are using SXNSX and don't have to coexist with anybody. Reported problems from users – zero. Hard to believe! Congrats on a great product." – Tom Uttormark

"I have never worked with a company as professional, responsive, and innovative as SuccessWare" – Glenn Butler

We installed Apollo in about 5 minutes with absolutely no problems. We are using databases created and maintained by a previous application written in Clipper using the SIx Driver, and have not missed a beat in our conversion to Delphi using Apollo. I thank you and SuccessWare for making my life easier" – Patrick Vandersluis

"I have written a package using Apollo and it would never have been possible without it. No way would I want to use the BDE, (and) it's great that you can use Apollo with the Delphi data components." -- Dominic Lee-Delisle

"You have a great product. Makes me, as a Clipper programmer, feel more comfortable with Delphi." -- Ronny Bolle

"I can't think of a more useful product than Apollo. But then again, I shouldn't even be telling you these things. In the consulting business, it's best to keep your money-making secrets to yourself."
-- Pat Buchanan

"Let me say that Apollo is a real first-class product. If it did nothing else but eliminate the need to distribute the BDE, it would be worth it's weight in gold. But it does that and provides fast, straightforward and extensive database management functionality to boot!"
– John Diehl

"Over the past week I've done my first demos of my applications using Delphi/Apollo to prospective customers. The customers loved it and we made one important sale that we didn't expect. The speed of the database access was important and the fact that we can install everything from one diskette." -- John Wright

"I think Apollo is a very good alternative to the BDE. Especially for developers such as myself that are developing shareware applications that need small distribution files. I want to continue development using Delphi and, as long as companies like SuccessWare continue to give us good alternatives, I will." – Bob Ludwig

"Just got Apollo about 2 weeks ago and already have forms at clients sites using legacy FoxPro tables that are shared by the existing FoxPro systems. Performs much better than ODBC." – Mark T. Butler

"Take my advice and take a S E R I O U S look at Delphi with Apollo from SuccessWare ... it's like having Clipper with all the hard work already done!" -- Tom McMinn

TApollo Methods:

While a great portion of APOLLO's functionality can be accessed via native TTable methods and properties, there are some features and capabilities within APOLLO that are not supported in Delphi's TTable. This additional functionality is contained, for the most part, within APOLLO's TApollo component and includes things such as optimized Xbase queries, single-order (.IDX/.NTX) index file handling, index scoping, database table, encryption, and even simple things that TTable does not support like getting the current record number.

Table / Structure Creation

CopyFile	Copies the current database to another file.
CopyStructure	Creates a database that has the same structure as the current database.
CreateExec	Creates a database in the work area set up via CreateNew and according to the field specifications defined with CreateField.
CreateField	Defines a field to be included in a new database.
CreateNew	Initializes a new work area.

Table Data Updates / Modifications

Append	Places a new, empty record buffer in Edit mode.
AppendBlank	Places a new, empty record buffer in Edit mode.
AppendFrom	Appends data from a source database to the current database.
Commit	Writes the contents of the record buffer to the current file position.
DBFDecrypt	Decrypts a database that has been encrypted at the file level.
DBFEncrypt	Encrypts all records in a database.
Delete	Flags the current record for deletion.
Pack	Removes all records marked for deletion from a table.
PutBlob	Stores a BLOB in a memo field.
PutRecord	Replaces an entire record with the contents of a defined record structure or character buffer.
Recall	Recalls a record that has been logically deleted.
Replace	Replaces the named field's contents in the record buffer with the given data value.
Zap	Remove all records from the database.

Data Retrieval

BlobToFile	Writes a BLOB stored in a memo file directly to a new disk file.
GetBlob	Retrieves a BLOB that was stored in a memo field.
GetByte	Extracts the first character of a field.
GetDateJulian	Extracts the contents of a date field as a Julian number.
GetDateString	Extracts the contents of a date field as a string formatted according to the date mask specified by SetDateFormat and the setting of SetCentury.
GetDouble	Extracts the contents of a numeric field and convert to a double value.
GetInteger	Extracts the contents of a numeric field as a signed integer value.
GetLogical	Determines whether a logical field contains a True or False value.
GetLong	Extracts the contents of a numeric field as a signed long integer value.
GetMemo	Extracts the contents of a memo field as a PChar and optionally format the memo with hard carriage returns and line feeds for printing.
GetRecord	Fills a defined and initialized buffer or record structure with the contents of the current record buffer.
GetString	Extracts the contents of any field (less than 256 bytes wide) as a string value.
GetStringEx	Extracts the contents of a character field as a PChar value.
GetTrimString	Extracts the contents of any field as a string and trim trailing spaces.

General Table / Workarea Information

Alias	Retrieves the database alias name assigned to the current work area when the database was opened.
BaseDate	Retrieves the date on which the database was last opened for read/write access.
BaseName	Extracts the DOS path and file name attached to the current work area.
Bof	Tests if a skip has been attempted that would place the record pointer before the first record in the file.
DBFilter	Extracts the current filter conditional expression.
Deleted	Determines whether or not the current record has been logically deleted.
Eof	Tests if a record movement function has placed the record pointer beyond the last record in the file.
FieldCount	Extracts the number of fields in the current database.
FieldDecimals	Extracts the number of decimals defined for a numeric field.
FieldName	Extracts the name of the nth field.
FieldNum	Gets number of named field in field array (relative to 1).
FieldOffset	Gets offset of named field in record buffer (relative to 1).
FieldType	Reports the type of the named field.
FieldWidth	Extracts the width of the named field.

GetBlobLength	Retrieves the length of a BLOB that was stored in a memo field.
GetScope	Gets the original value of a scoping string passed to SetScope.
IsEncrypted	Determines whether the current record or file is encrypted.
RecCount	Extracts the physical number of records in the current database.
RecNo	Extracts the current physical record number.
RecSize	Extracts the size of a record in the current database in number of bytes.
Select	Selects an existing work area and make it the current work area.
SetPassword	Defines an encoding key to be used in encrypting and decrypting data records in the current database.
SetTurboRead	SetTurboRead is used to speed up Apollo record i/o methods (specifically Skip and Seek).
WorkArea	Retrieves the database numeric work area identifier associated with a given alias name.

Table Navigation

Go	Goes to a physical record that occupies the nth position in the database.
GoBottom	Moves the record pointer to the last record in the file.
GoTop	Go to the first record in the file.
SetRelation	Defines a relationship between the current database and another database.
Skip	Skips forwards or backwards a specified number of records.

Index Creation / Manipulation

CloseIndexes	Closes all open index files attached to the current table object.
IndexClose	Closes the current index file.
IndexCondition	Extracts the conditional expression from the current index.
Index	Creates a new single-order (.NTX or .IDX) index file and make it the active order.
IndexKey	Extracts the contents of the xBase key construction expression used by the current index as a string.
IndexKeyField	Extracts the name of the first field in an index key expression.
IndexName	Extracts the DOS file name and path of the defined index.
IndexOpen	Opens an index file.
IndexOrd	Gets the current order number identifying the order work area.
IndexTag	Creates a new tag in the compound .CDX or .NSX index.
IndexType	Gets the current index or Tag type.
KeyAdd	Adds a key to the specified index order.
KeyData	Extracts the key data from the current index key in the current index and returns it as a string.
KeyDrop	Removes a key from the specified index order.
OrderRecNo	Retrieves the logical record number according to the position of the key in the current active order.
Reindex	Rebuilds all active orders attached to the current work area.
SetOrder	Selects an existing order as the controlling index order.
TagArea	Retrieves the index select area of the named tag within a compound index.
TagName	Retrieves the name of the tag associated with the passed tag area number.

Querying Data / Expressions

Count	Extracts the number of records in the database, respecting any active filter.
Empty	Determines whether or not the named field is empty.
EvalLogical	Evaluates a logical xBase expression and retrieves a True or False value depending upon the outcome.
EvalNumeric	Evaluates an xBase expression that returns a numeric value and convert the result to a double precision number.
EvalString	Evaluates an xBase expression and retrieves the result as a string.
EvalTest	Tests the validity of an xBase expression.
FilterAlias	Assigns an alias name to a field name in the database to be used in an upcoming

	FilterDlg call.
FilterDlg	Displays a modal dialog box that allows the user to create a custom filter expression visually.
Found	Queries the result of the last index seek made on the current work area.
GetQueryBit	Get the setting of a bit that maps to a record number in an existing bitmap.
Locate	Initializes or continues a record location command.
Query	Sets an ultra high speed filter.
QueryRecCount	Extracts the number of records contained in the current query subset.
QueryTest	Tests if a given xBase expression will result in an optimized query set or not.
Seek	Searches the current order for a supplied key.
SeekBin	Searches the current order for a supplied key. The key may contain binary zeroes.
SetFilter	Defines a subset of the current database according to the condition passed.
SetQueryBit	Sets a bit (on or off) in a query bitmap for the specified record number.
SetScope	Sets formal scope based on current index order expression.

Record / File Locking

FLock	Locks the current database.
Locked	Determines the lock status of a record or a file.
RLock	Locks the defined record.
Unlock	Removes record and/or file locks from the current work area.

Conversion

Descend	Converts a key string into a 2s complement representation.
SetTranslate	Automatically translates record buffers stored in the OEM character set to Windows ANSI.

Miscellaneous

Decrypt	Decrypts an encrypted string using the specified password.
Encrypt	Encrypts a string using the specified password.
MemDealloc	Deallocate memory allocated by GetMemo.
SetGaugeHook	Defines a window that time consuming operations may communicate with in order to inform the user as to the progress of the operation.
SysProp	Sets or Retrieves SDE system information.
Version	Extracts the name and version number currently in use.