BLAISE PASCAL MAGAZINE 85

Object Pascal / Internet / JavaScript / WebAssembly / Pas2Js / Databases
CSS Styles / Progressive Web Apps
Android / IOS / Mac / Windows & Linux

Blaise Pascal

Special Edition: DELPHI 25 Year

BLAISE PASCAL MAGAZINE 85

CONTENT

ARTICLES

ADVERTISERS

Stephen Ball
http://delphiaball.co.uk
@DelphiABall

Peter Bijlsma -Editor
peter @ blaisepascal.eu

Dmitry Boyarintsev
dmitry.living @ gmail.com

Michaël Van Canneyt,
michael @ freepascal.org

Marco Cantù
www.marcocantu.com
marco.cantu @ gmail.com

David Dirkse
www.davdata.nl
E-mail: David @ davdata.nl

Benno Evers
b.evers
@ everscustomtechnology.nl

Bruno Fierens
www.tmssoftware.com
bruno.fierens @ tmssoftware.com

Holger Flick
holger @ flixments.com

Primož Gabrijelčič
www.primoz @ gabrijelcic.org

Mattias Gärtner
nc-gaertnma@netcologne.de

Peter Johnson
http://delphidabbler.com
delphidabbler @ gmail.com

Max Kleiner
www.softwareschule.ch
max @ kleiner.com

John Kuiper
john_kuiper @ kpnmail.nl

Wagner R. Landgraf
wagner @ tmssoftware.com

Vsevolod Leonov
vsevolod.leonov@mail.ru

Andrea Magni www.andreamagni.eu
andrea.magni @ gmail.com
www.andreamagni.eu/wp

Paul Nauta PLM Solution Architect
CyberNautics
paul.nauta @ cybernautics.nl

Kim Madsen
www.component4developers

Boian Mitov
mitov @ mitov.com

Jeremy North
jeremy.north @ gmail.com

Detlef Overbeek - Editor in Chief
www.blaisepascal.eu
editor @ blaisepascal.eu

Howard Page Clark
hdpc @ talktalk.net

Heiko Rompel
info @ rompelsoft.de

Wim Van Ingen Schenau -Editor
wisone @ xs4all.nl

Peter van der Sman
sman @ prisman.nl

Rik Smit
rik @ blaisepascal.eu

Bob Swart
www.eBob42.com
Bob @ eBob42.com

B.J. Rao
contact @ intricad.com

Daniele Teti
www.danieleteti.it
d.teti @ bittime.it

Anton Vogelaar
ajv @ vogelaar-electronics.com

Robert Welland
support @ objectpascal.org

Siegfried Zuhr
siegfried @ zuhr.nl

Editor - in - chief
Detlef D. Overbeek, Netherlands Tel.: Mobile: +31 (0)6 21.23.62.68
News and Press Releases email only to **editor@blaisepascal.eu**

Editors
Peter Bijlsma, W. (Wim) van Ingen Schenau, Rik Smit

Correctors
Howard Page-Clark, Peter Bijlsma

Subscriptions (2019 prices)

	Internat. excl. VAT	Internat. incl. 9% VAT	Shipment
Printed Issue ±60 pages	€ 250	€ 261,60	€ 85,00
Electronic Download Issue 60 pages	€ 60	€ 65,40	———
Printed Issue inside Holland (Netherlands) ±60 pages	———	€ 200,00	€ 60,00

Subscriptions can be taken out online at **www.blaisepascal.eu** or by written order, or by sending an email to **office@blaisepascal.eu**
Subscriptions can start at any date. All issues published in the calendar year of the subscription will be sent as well.
Subscriptions run 365 days. Subscriptions will not be prolonged without notice. Receipt of payment will be sent by email.
Subscriptions can be paid by sending the payment to:
ABN AMRO Bank Account no. 44 19 60 863 or by credit card or Paypal
Name: Pro Pascal Foundation-Foundation for Supporting the Pascal Programming Language (Stichting Ondersteuning Programeertaal Pascal)
IBAN: NL82 ABNA 0441960863 BIC ABNANL2A VAT no.: 81 42 54 147 (Stichting Programmeertaal Pascal)
Subscription department
Edelstenenbaan 21 / 3402 XA IJsselstein, The Netherlands
Mobile: + 31 (0) 6 21.23.62.68 **office@blaisepascal.eu**

Member and donator of WIKIPEDIA

C - Q U E L

In need of Delphi support ?

Because our people almost breathe Delphi you can ask us anything about Delphi. If you're in need of some extra hands to speed up your project, if you want to port your project to the web, if you want to slim your fat client, don't hesitate to give us a call.

How can we help you ?

Contact us on +32 (11) 72 61 83 or mail us on info@cquel.be

Want to know more about us? See https://www.cquel.be

There is a Embarcadero Delphi MVP on the team and
we are tmssoftware.com certified technical partner for the Benelux.

s

embarcadero MVP Delphi®

tmssoftware.com

https://www.cquel.be

barnsten Locatie: Landgoed Jachthuis Beukenrode in Doorn (bij Utrecht)

BOUW JE EIGEN VCL APPLICATIES VOOR WINDOWS 10

Delphi VCL Essentials Training

We starten het jaar goed met een nieuwe driedaagse Delphi VCL Essentials training! Deze populaire training wordt verzorgd door trainer Danny Wind (Delphi MVP) en is inclusief zeer uitgebreid Nederlandstalig les- en oefenmateriaal.

Voor wie is deze training:

- Voor coders en ontwikkelaars, met enige programmeer ervaring in Delphi of een andere taal, die graag snel en goed met Delphi aan de slag willen.
- Voor Delphi ontwikkelaars die willen overstappen van een oudere versie naar de nieuwe 10.3 Rio versie en hun huidige kennis willen opfrissen.

De training begint met een overzicht van alles wat u met Delphi en de VCL kunt bouwen. Daarna gaat u de diepte in en leert u hoe u zelf VCL applicaties kunt bouwen en onderhouden. Hierbij komen ook zaken aan bod zoals werken met databases, vormgeven van applicaties, debuggen, touch en veel meer.

Wilt u ook alles uit uw Delphi omgeving halen en gebruik maken van de nieuwste functionaliteit? Schrijf dan nu in. De training wordt gehouden in Doorn (bij Utrecht) op 11 t/m 13 maart 2020.

In verband met de kwaliteit van de training werken wij met kleine groepen en is het aantal plaatsen beperkt.

Heeft u vragen over de training of over Delphi software? Neem dan direct contact met ons op via 023 542 22 27 of per email operations@barnsten.com

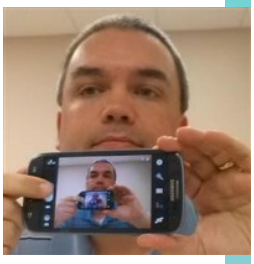
https://www.barnsten.com/nl/product/delphi-vcl-essentials-training/

Free Delphi Pascal Track on exercism

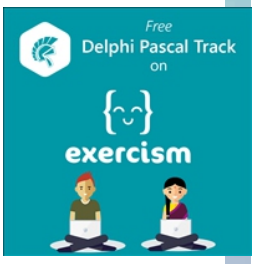
Jim McKeeth

One of our **MVPs, Ryan Potts,** is the originator and maintainer of the Delphi track on Exercism - a site dedicated making it easy for people to learn different programming languages. He was interviewed by the Exercism team previously **(https://exercism.io/blog/interview-with-ryan-potts)**, but I thought it would be interesting to interview him about what Exercism is, and how all of you can be involved.

- **What is Exercism?**
  How long has it been around, and how many students visit it?
  - A not-for-profit organization providing opportunity through programming education
  - The Delphi track had 3 new solutions submitted, 1 for mentoring.
  - See their about page for more **https://exercism.io/about**
  - It was originally founded in 2013 by Katrina Owen.
  - Each week I receive a mentoring update. The most recent update indicates that during the last week of 2019 2,152 people submitted 8,107 solutions across all the language tracks. Of those 2,339 were submitted for mentoring. 174 mentors gave feedback on 1,963 solutions.



- **What sort of programming languages are found on Exercism?**
  - There are 51, including our favorite, Delphi!
- What kind of exercises might someone find on Exercism?
  - There is the book-store exercise, my personal contribution. Read its description here: **https://github.com/exercism/problem-specifications/blob/master/exercises/book-store/description.md**
  - Bob -- The lackadaisical teenager **https://github.com/exercism/problem-specifications/blob/master/exercises/bob/description.md**
  - They run the gamut. You can see a complete list of all 134 (and counting) exercises in the specifications.



- **Do any of the exercises involve building Graphical User Interfaces or accessing databases?**
  What about accessing REST APIs?
  These are things that Delphi makes so much easier. Or are they all console applications?
  - At the moment all the exercises are console only. I have considered creating some Delphi specific exercises to utilize Delphi's ability to pretty easily build GUIs, just haven't had the time myself. This would be a great way other experienced Delphi developers to help.

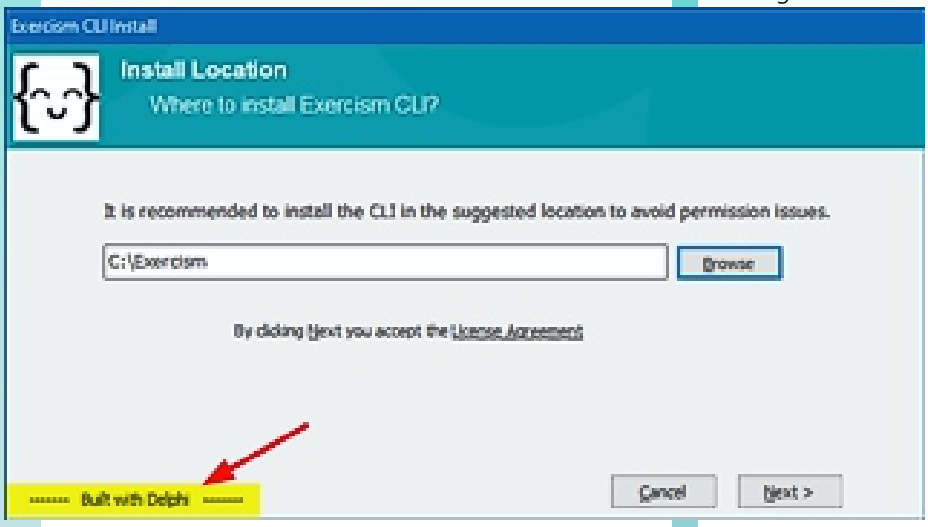
- **Does a student need any experience to start?**
  - No, but knowledge of the command line is essential.

- **Does it cost anything?**
  - No, it is free.
  - Exercism's core values **https://exercism.io/values**
  - Getting Started **https://exercism.io/getting-started**
  - FAQs **https://exercism.io/faqs**
  - Info Page for Delphi Track **https://exercism.io/tracks/delphi**

- **What sort of tools does a student need to get started?**
  - Each language track has set up instructions to let you know what you need and where to find it. For example with Delphi you need to install Delphi. There is also a command-line tool that all the tracks use.

- If a student is completely new to Delphi what do they need to do to get started.
  - Once someone has signed up for the Delphi track they will find instructions on how to set up their system **https://exercism.io/tracks/delphi/installation.** (They are a bit dated, a few versions of Delphi have been released since I wrote these.)
  - Other links are also available from the same location for more Delphi related resources.

- **Is it true the Exercism Windows CLI installer is written in Delphi?**
  - Exercism's Windows CLI Installer is open source and written entirely in Delphi. It automatically determines the correct version (32/64 bit) of the Exercism CLI the student needs then downloads the CLI, extracts it, and puts it in the right place. The installer has been downloaded 15,000 times, which I'm pretty proud of!



- **Is Exercism only for new developers or are there things experienced developers might learn there as well?**

It is not only for new developers, but is probably geared towards someone with only a little experience, knowledge of how to operate a terminal (command prompt) interface is essential.

- **How is Exercism similar to a code kata? Code Kata**
  - It essentially IS a repository of code katas; with added benefit of having someone knowledgeable in the language you are working in review and critique your work. The term code kata is a concept first used by **Dave Thomas,** co-author of the book The **Pragmatic Programmer** as a nod to the Japanese concept of kata in the martial arts. A code kata is an exercise in programming which helps programmers hone their skills through practice and repetition.
- **Beyond being a student on Exercism how else can people be involved?**
  - To get an idea of what should be contained in the PR it is a good idea to view one of the other exercises in the /exercises folder of the /exercism/delphi repo. The maintainer (me) will review it, offer suggestions for changes and ultimately I can then merge the PR.
  - New exercise ideas or contributions are always appreciated.
  - Working with the maintainer(s) of the tracks you frequent is also appreciated.
  - See the contributor guide
  - Just create a pull request (PR) on the Delphi exercises track to contribute additional exercises.
  - For a first time contributor it might be a good idea to find an exercise from /exercism/problem- specifications that is NOT yet implemented and write a test suite for it. It can be done blindly, by simply writing code based on the provided JSON, or by translating the implementation of that exercise that might exist in another track.

- **How does someone become a mentor?**
  - Don't need to be a student to be a mentor, but unless you have done a lot of contributing to the project being a student is the next best way to start to get an understanding of how things work.
  - I did a few Ruby exercises and decided it would be really cool if Delphi was offered here. I was quite frankly frustrated at how many online coding sites are out there and none I could find offered Pascal.

Researching Exercism a little further I learned that they welcomed the addition of new language tracks if you are willing to put in the time to set it up and maintain it.

- First need to sign up as a student and must submit at least one solution.
- Become a mentor

- **Is the Delphi Exercism track looking for more maintainers?**
  - Certainly. I have been doing it on my own since 10/2016. It isn't hard but there are some interesting things going on in some other tracks that I have not had the time/energy to pursue on my own.
    - Creating a test generator -- Converts exercise definitions (in JSON) to native language (Delphi for example). I have been writing the test runners by hand. A test generator could be written in any language, but traditionally track maintainers write their generators in their language.
    - Some tracks have started to implement auto-mentoring. Some early exercises in a track are very simple and the number of ways to implement a solution are limited and it ends up consuming a lot of mentor time to review these same exercises over and over.
      The auto-mentor is programmed to look for familiar patterns and make some canned recommendations. Ultimately falling through to a human mentor if a good expected response cannot be provided.
    - Having a second maintainer also is a good backup too.
    - Learn how to become a maintainer **`https://exercism.io/become-a-maintainer`**

I've signed up as a mentor and submitted my first pull request on the Delphi exercises! Looks like a lot of fun.
If you know someone who would like to learn to program this is a great resource.
Or maybe you would like to become a mentor or maintainer too!

By the way, Ryan is one of the champions on the new LearnDelphi.org project!

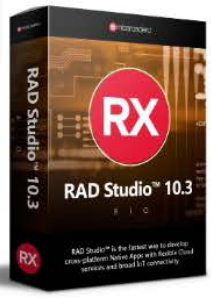
Jim Mckeeth

barnsten

embarcadero

Shop your 10.3 RIO here. Now with 25% Anniversary discount!

From
€2.999,-



RAD Studio 10.3 Rio

RAD Studio 10.3 makes significant improvements for C++ developers including C++17 Win32 support, faster math performance in Win64, improved code completion, debugging for optimized builds and new libraries.

Shop RAD Studio

From
€1.699,-

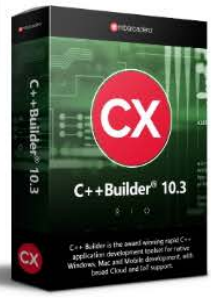
Delphi 10.3 Rio

RAD Studio 10.3 makes significant improvements for C++ developers including C++17 Win32 support, faster math performance in Win64, improved code completion, debugging for optimized builds and new libraries.

Shop Delphi

Most popular

From
€1.699,-



C++Builder 10.3 Rio

RAD Studio 10.3 makes significant improvements for C++ developers including C++17 Win32 support, faster math performance in Win64, improved code completion, debugging for optimized builds and new libraries.

Shop C++Builder

https://www.barnsten.com

I'm Anders **HEJLSBERG** and I'm a Microsoft technical fellow.
My first sort of real introduction to coding was probably in high school.
The high school I attended was one of the first high schools in Denmark to offer the students access to a computer and this is before pcs or anything like that
It was actually technically what we'd call a mini computer and I think it was an HP 2100

It had 32 K a ferrite core memory you could literally open it up and see the little ferrite cores and paper tape reader
A teletype you know and then we got like a real CRT terminal and I learned to program that thing in Algol and it was kind of like for the first two weeks the teachers would teach us - and then we would teach the teachers
because you just had to discover it all for yourself by trial and error.

So when I started at the TechnicalUniversity I met another fella- actually we were on a trip for all the first-year students.
We were playing cards and he was loosing some money and so and he was writing some IOU (I owe you).
So I had to get to know him and we ended up starting a small company together.
We had the very first computer store in Copenhagen where you could walk in and actually buy a computer.
Back then there was mostly kit computers that you have to put together yourself.
But then we also ended up selling Apple tubes and TRS 80s you know .

Later I did a lot of just programming on these things and in the beginning you know, your sort of programming what's there, like in Basic and:
That get's boring pretty quickly and then you start writing machine code.

and then we would teach the teachers

Guess that's when you could really get down to the middle and then you sort of start experimenting with writing stuff for the machine like extensions to the ROM basic or a small operating system or an on-screen editor.
And then, eventually I wanted to write an Algol-Compiler
(ALGOL short for "Algorithmic Language") is a family of imperative computer programming languages, originally developed in the mid-1950s, which greatly influenced many other languages and was the standard method for algorithm description used by the Association for Computing Machinery (ACM) in textbooks and academic sources for more than thirty years.)
and then my friend was going:: No, No,No -there's this other thing called **Pascal** - you should check it out.

It sounds really cool and we ended up writing a subset of Pascal with a little on-screen editor and a little runtime package.
It was only 12 K of machine code and it fit into a ROM.
So you could yank out the Pascal and then when you turned your computer on,

you were sitting a little on screen editor and you had Pascal.
Now it was a **subset of Pascal** but of course then that grew into fuller implementation for CPM you know.
At that time eventually for PC Dos - I think around - 82- maybe, maybe 83,1983.

We ended up meeting up with some company in Denmark they wanted to get out of the hardware business and into the software business and they had written a couple of programs.

One was called **word index** which could create table of contents and, and index generator for **Word Star** - which was this popular text processing system at that time..

They were writing in Pascal and they were using a very crappy Pascal compiler..
And then - you know - we were looking at each other,: **we actually have a very good Pascal compiler you should check it out**.
- It's this and that fast - we were telling them.
And it had like an on screen and they didn't believe a word of it.
And so we're sitting here: „try a copy"
and then they came back:
Oh. This is amazing - we should do business together - and then we ended up actually doing a royalty contract.

That's how **Borland** ended up producing **Turbo Pascal**. It was our Pascal compiler you know rebranded as **Turbo Pascal**.

Obviously that the fact that it took the traditional edit compiler run debug cycle and squeezed it from you know sometimes- It measured in minutes if not hours you know.

Depending on what kind of compiler you were using - uh - some of them you one, and pass to with a compiler.
You actually have to swap disks with pass one and pass to with a compiler - you know.
I mean - we just brought that down to like seconds and it was instantaneous and the interactive feel of it made compiled interactive languages.

Languages feel like they were dynamic or interactive languages right, and that obviously is just in retrospect the right way to do it.

So that that clearly was one of them right.
I mean the other was the fact that it would run on any piece of hardware.
that was that contemporary - at the time right it was only **32 K Turbo Pascal.** 32k of machine code so even on an 8-bit machine there was a memory left over for you to write programs and then I think finally it was the price.
At the time products like that typically sold for 500 bucks and we lowered the price by an order of magnitude down to 49 95 right and at that point it's like wow.
Heck there's really no point in pirating it you know for 40 to 49 bucks. It's worth it just to get the manuals that explain how it works dude jr.
In combination - I think were quite new and different at the time.

They'll find many ways. That was really just sort of a successor to Turbo Pascal.
I think I would say - with the Delphi project -
I learned personally how to be a better team player.
I started out just being a perfectionist that wanted to do everything myself. So the entire compiler and runtime library was written by me for Turbo Pascal.

There was an editor and the command driver for a few other people that worked on the editor and and the command driver for the ID itself but obviously that doesn't scale you know.
And then and so eventually you got to have a team to help you out.
I was hired to be the architect of Microsoft's Java development tools. At the time there was a really big push inside Microsoft to do Java and - you know - like even build applications in Java or maybe even build an entire OS in Java or

you know - I mean - it was like that time when when Java was just taking the world by storm and in the early internet days.

But Microsoft didn't have a Java development tool at all though you know.
The closest was their **visual C++** development environment and, and that was really not at all the right fit and and so I, I was brought on to spearhead that.

But it very quickly turned into a political battle between Microsoft and Sun and, and that the product that we had built **Visual C+ +.**
Or it was the 6-O version ended up having - you know a judge in San Jose enjoying it and requiring us to put up warning dialogues about Microsoft proprietary language extensions.

So in a sense that also was the genesis for what came next which was **DOTNET** and **C-SHARP**
Because **Microsoft** as a company realized that it's not gonna work for us -
and for our customers to build our future platform on top of something that we have licensed from some other partner .
We knew from our customers that that is too hard to write code in **C++** and just not enough programmers could do it.
We did have at the time **VISUAL BASIC** but that wasn't considered a grown-up full-blown programming language and it didn't compile to native code
And so there was a really strong sense that that's something with the ease of use of **Visual Basic** but with the power of **C++** was really what programmers wanted.
So all of those take them together and that was really sort of the genesis for for building the the **.net** framework and, and of course on top of that then a programming language called **C sharp**.
it turns out that at that time **JavaScript** had all sorts of issues you know there were no classes.
There were no modules and there was no static type system in **JavaScript.**
So it was very, very hard to write large apps in **JavaScript** and also the tools for **JavaScript** at the time were really terrible.

I felt like you were just coding in **notepad** - you know.
I mean there was no statement **`completion`** and **`code navigation`** or no notion of **`statically checking`** your program before you run it.

I and and so teams were actually choosing to write in a different language and cross compiled to **JavaScript.**
Just treat **JavaScript** like an IL (Intermediate Language) you know.
I mean and and - I felt like wow - wel I mean if you really want to be best, a breed best possible JavaScript development environment.

What if we instead we could fix some of these issues you know what let's try to figure out what are the issues what are the problems and why are the tools crappy and what can we do to make him better and that was in a sense the genesis for, for, for typescript.

We started out just **thinking** we were doing open source - we were doing everything the way we used to.
Do it with our internal processes and then we would sort of lop the source code out in a repository and then when people logged issues we'd scraped the issues and put them back into our internal issue tracking system.
And, and that was sort of that, but that's not - well - that's **technically** open source - right?
But it's not open devlopment.

Ultimate where you're actually doing your development in the open as well and we switch to that after a couple of years and I think that changed everything really because our development process now is you know we are so close to our users.
I mean they're literally there.
Every day and the entire developer team is accessible and on **github** - you know that whole workflow allows you you to set yourself up to to be super super fast in addressing issue

So, so if if someone reports a high priority issue, will typically have a fix during that day and and it the fixed version will be in the nightly build.
That is so profoundly different you know from from how we used to do our work
And it's some - it's a much better way - to do it. It's you know, it's better for everybody and it's better for the team too - **because it's so rewarding to be that close to your users.**

Development tools are like enormous.
The place where programmers invest an enormous amount of time in their programming language and in their developer tool and because they have such a big investment and they are in the same manner **super passionate** about these tools.
And some when you delight them there's so much enthusiasm and ultimately the rewards you get from seeing people's and enthusiasm and using the tools that you've created it is just very fulfilling.
You know that that is at the end of the day which which after 35+ years still makes it worth worthwhile for me to do it you know is this just that that people are so excited about it and I just love that you know...

Don't be afraid you to - don't let people tell you that it can't be done.
You know a lot of people will go: „oh no we tried that decade it's not possible"
you know - well - I mean that means it might not be possible for you. But, but maybe someone else can do it. You never, **you never know**, right, I mean, and we've always sort of see in retrospect that: Oh you were thinking about it wrong. right and: or there's this simpler way that you could have gone about it, or so.
So always be curious you know and don't, - don't take the dogma for given.

Anders Hejlsberg, is a prominent Danish software engineer who co-designed several popular and commercially successful programming languages and development tools. He was the original author of Turbo Pascal and the chief architect of Delphi. He currently works for Microsoft as the lead architect of C# and core developer on TypeScript.

Hejlsberg was born in Copenhagen, Denmark, and studied Electrical Engineering at the Technical University of Denmark. While at the university in 1980, he began writing programs for the Nascom microcomputer, including a Pascal compiler which was initially marketed as the Blue Label Software Pascal for the Nascom-2.
However, he soon rewrote it for CP/M and DOS, marketing it first as **Compas Pascal** and later as **PolyPascal.**

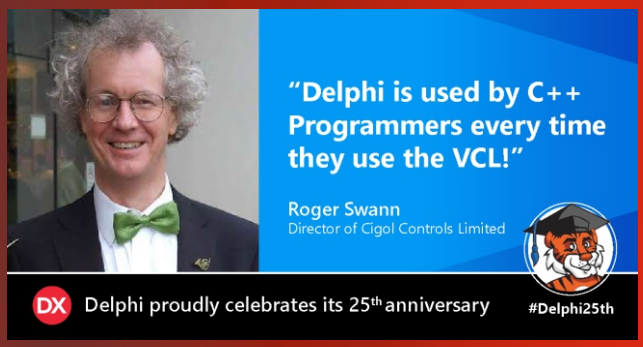
Later the product was licensed to Borland, and integrated into an IDE to become the Turbo Pascal system. Turbo Pascal competed with PolyPascal. The compiler itself was largely inspired by the "Tiny Pascal" compiler in Niklaus Wirth's "Algorithms + Data Structures = Programs", one of the most influential computer science books of the time.

In Borland's hands, Turbo Pascal became one of the most commercially successful Pascal compilers. Hejlsberg remained with PolyData until the company came under financial stress and in 1989 he moved to California to become Chief Engineer at Borland.
During this time he developed Turbo Pascal further and became the chief architect for the team that produced Delphi, which replaced Turbo Pascal.

The interview with Anders Hejlsberg isoriginated here: .EXE Interview with Anders Hejlsberg on Delphi (1995)
https://delphi.embarcadero.com/interview-with-anders-hejlsberg/

https://www.theopenforce.com/2020/02/anders-hejlsberg-delphi-1995.html
It was transcribed by BPM from the You tube version of the video.
Extra info can be found at:
https://delphi.embarcadero.com/


"I started using Delphi in 2001 and have completed many successful projects."
Dalija Prasnikar
Software Developer
DELPHI 25 YEARS
Delphi proudly celebrates its 25th anniversary


"Delphi is used by C++ Programmers every time they use the VCL!"
Roger Swann
Director of Cigol Controls Limited
Delphi proudly celebrates its 25th anniversary
#Delphi25th


"Work with Delphi for over ten years and teach many Delphi courses in Brazil."
Dion Mai
Software Development Manager
Delphi proudly celebrates its 25th anniversary
#Delphi25th


"Delphi fan since Delphi 2 and still loyal!"
Didier Cabalé
Embarcadero Delphi MVP
DELPHI 25 YEARS
Delphi proudly celebrates its 25th anniversary


"Delphi certainly changed my life direction, and still impacts me everyday."
Roy Woll
President Woll2Woll Software
DELPHI 25 YEARS
Delphi proudly celebrates its 25th anniversary


"I started using Delphi in 2001 and have completed many successful projects."
Ahmet Nuri
General Manager At Camart Arge
Delphi proudly celebrates its 25th anniversary
#Delphi25th


"I've been using Delphi since version 3 from 1998."
Alister Christie
Consultant / Developer / Trainer
DELPHI 25 YEARS
Delphi proudly celebrates its 25th anniversary


"Delphi is one of my favorite programming languages!"
Scott Hanselman
Developer
Delphi proudly celebrates its 25th anniversary
#Delphi25th

Raize Software, Inc.

"My Delphi story goes way back to the days of Turbo Pascal. I fell in love with the language and tool back in college. My first programming job was at Argonne National Lab as an undergrad where I rewrote a data acquisition system in Turbo Pascal for the Physics Dept. Unfortunately, after graduating, I worked for several different companies, but none of them allowed me to code in Turbo Pascal.

However, I continued programming in Turbo Pascal on my own and I started writing articles for PC Techniques magazine. This lead to Jeff Duntemann offering me the opportunity to write a regular column for the magazine called Blazing Pascal. I am grateful to this day for Jeff giving me this opportunity, because shortly after I started the column, Jeff connected me to Borland about some cool new tool they were building.

I was also very fortunate to be allowed to publish the first Delphi article, "Introducing Delphi 95" in September 1994. I continued to write about Delphi in every PC Techniques issue that followed (including the rebranding to Visual Developer Magazine).

I was amazed at the feedback I received from those early articles. So much so that I decided to go into Delphi consulting. On Feb 7, 1995, a week before the official launch of Delphi, I started Raize Software Solutions. I found a client almost immediately, quit my job, and started training a group of Smalltalk developers at First Chicago Bank how to use Delphi.

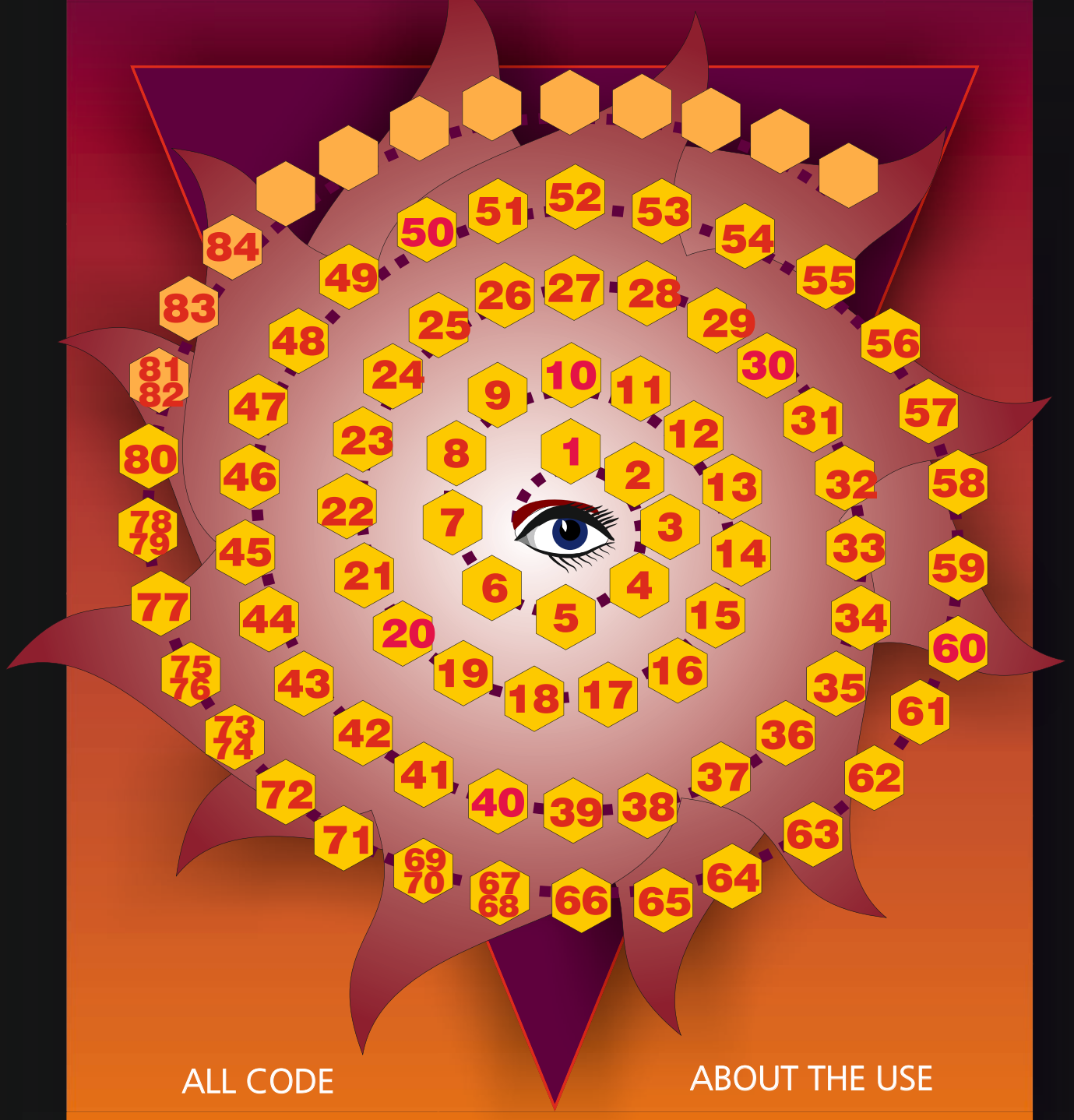
Shortly after I started, I was presented with a serious programming challenge. The developers at the bank understood object-oriented programming well, and they wanted to isolate their business rules from the user interface, but they also did not want to abandon Delphi's data-aware controls, which is what another consultant was proposing. This was unacceptable to the VP in charge of the department and the other consultant's contract was quickly terminated.

I got the call on a Friday that I needed to come up with a solution. I was excited that I was going be able to bill more hours, but I was freaking out. I had just started my consulting company. I hadn't even been able to submit an invoice for the training I had performed. Now I had to come up with a solution to a very complex problem very quickly, or my consulting career may quickly come to an end.

That weekend I created what I called Data-Aware Business Components. It was a huge hit with the bank. After that, I never looked back. That summer, my first book, Developing Custom Delphi Components was published, and I was selected to present some Delphi sessions at the 1995 Borland Conference.

The success of the book lead to the creation of Raize Components and CodeSite. I continue to use Delphi almost every day, and I still write Delphi articles/posts and present Delphi sessions at conferences. I consider myself very lucky, indeed."

LIBRARY 2020



ALL CODE

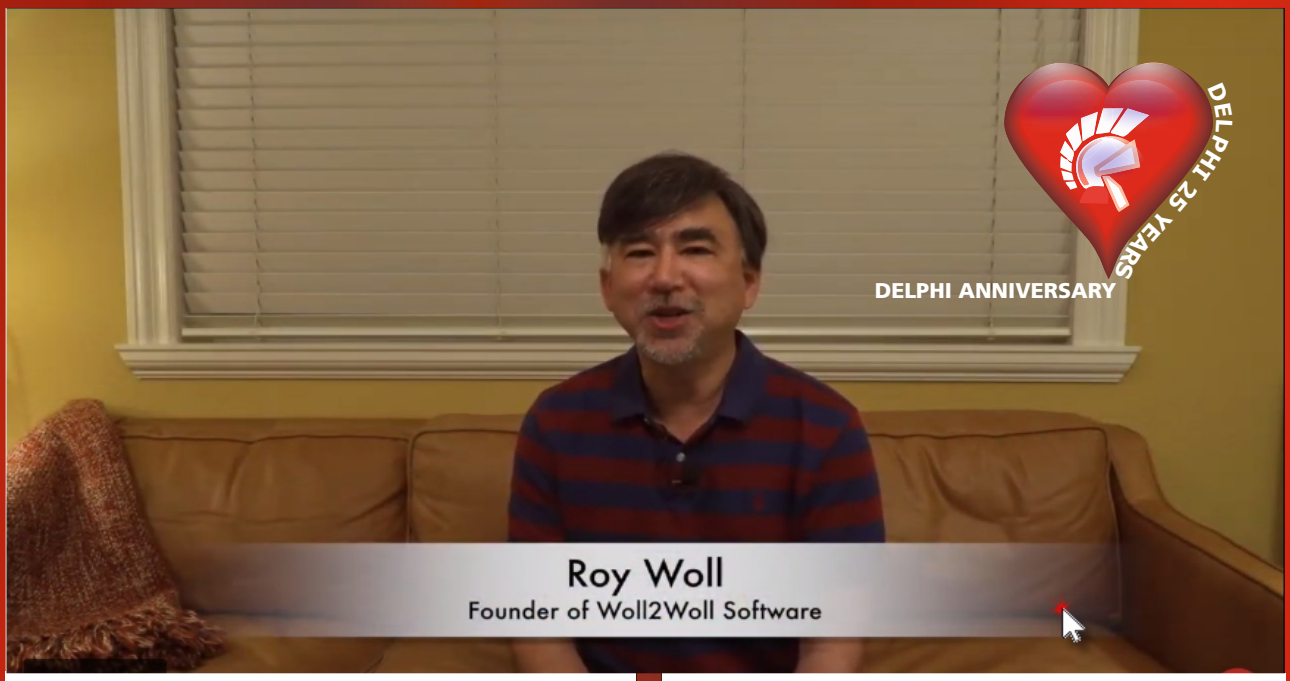
ABOUT THE USE

BLAISE PASCAL MAGAZINE

ALL ISSUES IN ONE FILE

CONGRATS FOR 25 YEARS OF DELPHI

The Lazarus Factory



"Hi. I like to introduce myself. I'm Roy Woll, and I am president and founder of Woll2Woll Software. I have been around Delphi for all 25 years and even before. To give some background,, I like to start with some personal history before Delphi was launched to give you some context of my life around that time.

In 1993 I was working at a corporation who used VAX/VMS systems to develop software to control telephone testing equipment. The job was ok, but not too exciting. At that time, I was much more passionate about Borland's lighting fast Turbo Pascal compiler and Paradox for Windows. I loved the revolution happening in the personal computer world. I even applied for a job at Borland at one point since I wanted to be part of this paradigm shift in development. Unfortunately after my phone interview with them, I did not hear back.
I thought I aced all their C++ questions, but maybe I got one wrong, or maybe they were looking for someone who was already using their tools in their day job. For me it was just an outside the work hobby, but I loved using the tools.

So since Borland didn't hire me, it was time to launch my own company Woll2Woll Software. We created add-ons for Paradox for Windows which included our documentation tool named ezDoc, and our library ezDialogs. They both were very well received, and our company became known in the Paradox world. Both of these products used Borland C++ to create Dlls that were integrated with ObjectPAL, the Paradox for Windows language. ObjectPAL was not quite up to creating these kinds of tools on its own, since it was mostly just a database language without much GUI language support.
About 18 months later, I was invited to a Delphi pre-launch in Scott's Valley. It was an exciting time with an in-depth presentation by the team. It was to be a full blown framework for windows development and database access.

It was a dream come true for me, as now I did not have to use two different tools (Paradox and C++) to do the job. Delphi could do everything and do it so naturally with its new VCL library. I was literally blown away. It compiled almost instantaneously and the programs ran incredibly fast. I immediately starting working with the beta, and trying to replicate the capabilities of Paradox for Windows. It became apparent to me that it needed more, such as a more advanced grid, better database support such as true database filtering, lookup fields, validation masks. So I began working night and day on our new product InfoPower.
Delphi came out and we launched about a month or two afterward. I thought my Paradox tools did well, but I had no idea the energy that was to be the Delphi community and InfoPower really took off. It really filled the void of the Paradox for Windows programmer who didn't want to sacrifice functionality or ease of development.

Since then of course Delphi kept getting better and better. It changed hands a few times to different companies but each new version brought excitement. More recently Delphi has introduced another fantastic new library named FireMonkey. Just as the VCL made windows development natural and easy, FireMonkey does the same for other platforms and devices. With the same excitement we created our new library FirePower about 5 years ago, to work with this new framework. Just as I was thrilled to develop for Windows 25 years ago with a better framework, I can now again for all the latest computers and mobile devices with the FireMonkey library.

I want to congratulate the Delphi team, the pioneers who started it and the community and leaders who keep it going strong. It certainly changed my life direction, and still impacts me everyday. If there is one word that exemplifies delphi for me as a programmer it is the word JOY. Thank you Delphi!"

BORLAND MUSEUM – WHY THE NAME "DELPHI?" PAGE 1/2

by Danny Thorpe

"Delphi" started out as a beta codename for a closely guarded skunkworks project at Borland: a next-generation visual development environment for Windows based on Borland's Object Pascal programming language.
The codename hatched in mid 1993, after the development team had been through about 6 months of deep research, proof-of-concept exercises, and market analysis.
Members of the then Pascal development team were hanging around R&D Manager Gary Whizin's office brainstorming clever codenames to use for the new product.
It was not a large office, but it was not a large team either – about 10 people between R&D, QA, Pubs, and Marketing.
It would have been odd not to see Anders Heilsberg, Chuck Jazdzewski, Allen Bauer, Zack Urlocker, Richard Nelson, myself, and several other regulars jawing away on some topic or another in Gary's office.
For the codename jam sessions, there may have been some overflow into the hallway.

Borland has a long history of "unusual" codenames, some with catchy slogans or backgrounds that tie the odd name to the market or product focus.
A codename should have no obvious connection to the product, so that if an eavesdropper overhears the name in conversation it won't be too obvious what product is being discussed.
The difference between an everyday disposable codename a great codename is the pithy passphrase behind it.
The most memorable for me was the codename for Quattro Pro 4.0: "Budda". Why?
It was to assume the Lotus position!

So we were sitting in Gary's office, kicking around weird and wacky codename possibilities.
The strategic decision to make database tools and connectivity a central part of the new Pascal product had been made only a few days before, so Gary was keen on having a codename that played up the new database focus of the proposed product, and of its development team.
The database shift was no small matter – I remember having grave reservations about "polluting" the Pascal tools with database arcana that took me almost a year to shake off.

It was a big gamble for Borland, but it was very carefully measured, planned, and implemented. In hindsight, making Delphi a database product was exactly what was needed to break Borland's Pascal tools out of the Visual Basic – C++ market squeeze play and set Delphi head and shoulders above traditional Windows development tools.

Gary kept coming back to the codename **"Oracle",** referring to SQL connectivity to Oracle servers. "Oracle" didn't fly with the group, though.
Aside from the obvious confusion with the same-name company and server product, the name itself implied server stuff, whereas the product we were building was (at that time) a client building tool, a way to talk to Oracle and other servers.

How do you talk to an oracle?
"The Oracle at Delphi" was the word association that popped into my head. So I offered up "Delphi": If you want to talk to [the] Oracle, go to Delphi.

The suggestion wasn't an instant hit.
It's an old name, an old place, a pagan temple in the ruins of a dead civilization.
Not exactly an inspiring association for a new product!
As some press articles later noted, the Delphic Oracle was particularly infamous for giving out cryptic or double-edged answers – not a great association for a data management tool.

Asking a question of the oracle was free to all, but having the oracle's answer interpreted and explained (compiled?) cost a pretty drachma. (The marketing guys liked that part)

Overall, though, the "Delphi" codename had a classier ring to it than the sea of spent puns that littered the room.

Pascal is a classic programming language, so it just felt fitting somehow to associate a Pascal-based development tool with a classical Greek image.
And as Greek mythologies go, the temple at Delphi is one of the least incestuous, murderous, or tragic ancient Greek icons you'll find.

We went through a lot of codenames during the development of that 1.0 product, coining a different codename for each press or corporate briefing of the beta product.
This was an effort to limit rumors and allow us to track the source of leaks.
The last thing we wanted was for you-know-who to get wind of what we were up to.
Through all these disposable codenames, the Delphi codename stuck.
Towards the end of the development cycle, marketing started using the Delphi codename across all prepress and corporate briefings, and as the codename for the final beta releases.
That got the rumor mills talking to each other, and the tools industry was abuzz with talk about this secret project at Borland codenamed "Delphi".

J.D. Hildebrand wrote a whole editorial in Windows Tech Journal about the "Delphi buzz" months before the product was launched. (paraphrased: "I can't tell you what it is, but I can tell you this: Delphi is going to change our lives.")

When it came time to pick a retail product name, the nominations were less than inspiring..

The "functional" name, a name that describes what the product actually does and is therefore much easier to market and sell, would have been **AppBuilder.**

This name actually still appears in some IDE internal class names, such as the class name of the IDE main window. (R&D caved in to the functional name pressures and set about implementing it early)
But AppBuilder didn't light up people's imagination. It didn't work well internationally – functional names are only functional in their language of origin.

Thankfully, a few months before Delphi was scheduled for release Novell shipped their own product called Visual AppBuilder. There was much rejoicing in the Borland halls, for at last the "AppBuilder" debate was laid to rest. With the functional name taken out of the running, suggestions started coming from all quarters to use the Delphi codename as the product name.

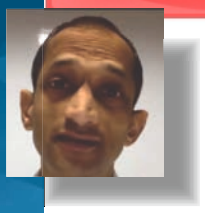
Delphi wasn't home-free yet.
The lead marketing person had legitimate concerns about the extra effort that would be required to build name recognition in the marketplace for an "iconic" (opposite of functional) product name, so he requested a vote of the development team.
There was only one vote against (guess who?).

Much to our chagrin, someone came to the conclusion that the development team's views were not an accurate representation of the marketplace ("sample error" was the phrase I heard), and pressed for a survey of the beta testers.

When that poll didn't produce the result he wanted, the survey was broadened again to include Borland's international subsidiaries, press, market analysts, stock analysts, corporate accounts, software retailers, and probably a few K-Mart shoppers.
It became a bit of a comedy: the harder people tried to dismiss "Delphi" for the product name, the more it gained support.

"Delphi" has a classical ring to it. It has a consistent meaning/word association worldwide in all languages.
It has no embarassing vulgar slang meanings in other languages (that I'm aware of).
Most of all, the marketing guys had done a marvelous job of building up market anticipation and buzz around the "Delphi" name.
The market was primed and ravenous for this thing called "Delphi".

And that, boys and girls, is how the Delphi product got its name.



"I started programming in Delphi with Delphi 1 sometime at the end of 1995.
And with Delphi came the seriousness to do something serious. It was really easy to get started with and actually get things done. I should add that before starting Delphi, I had programmed a bit in TurboC++ with the use of the TurboVision library and had also created a few components in TurboVision. But really, I started to understand OOP better when I started to use Delphi – the need, the use, and the usefulness of OOP.

My first major project with Delphi, which I wrote along with my cousin, was with Delphi 1 and was a store keeping software for my father's machine tool manufacturing company. This was in 1996. It did a lot more than just inventory management. It even assisted with production planning as it kept the bill of materials of the products manufactured and generated complete reports on what raw materials to procure based on a given customer order for the product the company was making. The users, the store keepers, really liked it.

Then with my friend I did several projects for different companies.
Among these projects was a customer management system, a help desk management system and a medical laboratory management system.
All these projects were built in Delphi 3.
The projects came on their own. We didn't go looking for them.

Towards the end of these projects, I developed a set of data aware lookup controls with all the advanced features such as auto-complete, multi column drop down and much more.
The product was called gplookup pack and it was my first commercially available component set.
A little later, one of the customers for whom my friend and I had built a crm system, wanted to be able to send out reports by email to their field staff around the country.
The requirement was that the reports should be viewable without having to install any additional software.
This I'm talking 1999 and converting the report to HTML came up as the choice. We used QuickReport at the time for reports. I looked at the export capability of QuickReport and saw that it really did not produce a useable HTML file of even simple reports, and we had reports which had multiple grouping levels.
I decided to write one and there begins the history of the company and what we work on now.
I wrote an export filter for QuickReport to convert to HTML and it produced a WYSIWYG output of the report.
The customer was happy. They could now email reports to their field staff.

I realised this converter is a component that every Delphi developer using QuickReport will need. Spent about 3 (continuous) days, rolled it into a proper component with properties to configure the output. Wrote a user manual, put up the component on my site, announced in the newsgroup and went to bed. Wake up in the morning and there's a sale for it.

So, documents became the area the company I founded would work on. To fast forward the story… Wrote a new product after that that include PDF and many other formats. Then we did bigger products in that area to arrive at where we are now.

We now have Gnostice Document Studio for Delphi, which support VCL and FireMonkey and does a lot of what the earlier products did and more.

So, Delphi really helped me find a business line in software and found a company. We have diversified since then, but continue to support and love working on our Delphi products."

IN THIS ISSUE - 14TH OF FEBRUARY 2020 WE ARE CELEBRATING THE 25TH ANNIVERSARY OF DELPHI.

You were there when the Delphi 1 was released. How did that go at that time?

Yes, I worked for Borland at the European Office in Amsterdam and a very innovative development environment would be released.

The code name of the product was Delphi.

Even before it was officially released on the market, there was plenty of speculation about Delphi and the name was so well known that eventually the code name became the official product name.

It were crazy times. Hundreds of people pre-booked their orders to be the 1st to have Delphi in house.

How was it at the Borland office?

The office was located in Buitenveldert (Amsterdam). It was Borland's European headquarters where the support department (where I once started) was located.

But also sales, marketing and e.g. the translation department. The manuals were still translated into different languages at the time.

We were a young group of people and always said: this is not work but a life style. I look back on a great time, with an international team of people, good atmosphere and the sky was the limit.

How did the logistics go at that time?

Delphi was shipped in a large box with a lot of floppy disks, manuals, flyers etc.

In the Netherlands, the products were stored in a warehouse in Amsterdam from where they were shipped to the Benelux customers.

It was important to make a good forecast to prevent the stock from running out.

But you also needed to be careful when a new release was expected that not too much stock was left.

And the orders, customer contacts etc?

It went completely different in those days.

We had no internet!!

We advertised in magazines and we sent out letters to companies with an order form.

They could send this back by mail or by fax.

The fax was continuously receiving orders.

Also during the evening.

Every morning I walked with my arms full of envelopes from the reception to my desk.

The contact with customers was by telephone and via large events.

And just like now, we had very diverse group of customers and users.

From students and self-employed people, to software development companies and multi-nationals.

How where new versions announced?

New releases were always announced during events. The new versions and functionality were presented to the developers during these events. This was the only way to provide detailed information about the tools. Many times Philippe Kahn or David I came over as evangelist. And there was always a big party at the end.

What is the story behind CodeGear?

In 2006 Borland decided to split up its product lines. They now also had an ALM product line. To ensure that the Developer tools such as RAD Studio, Delphi, C ++ Builder and InterBase continued to receive the attention they deserved, the tools were placed in the separate CodeGear division. This division was sold to Embarcadero in 2008 because the products no longer fitted into the Borland strategy.

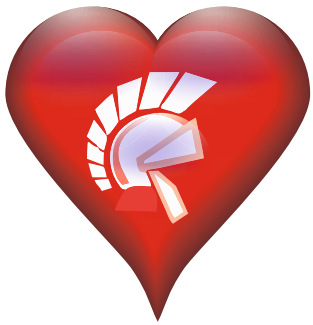
What happened after the takeover?
Borland / CodeGear decided to shut down the European office in the Netherlands.
Together with a few other colleagues, we decided to set up the **Barnsten** company.

Barnsten became the agency for the CodeGear Developer Tools in the Benelux and distributor of various other software suppliers.
Awesome!
I was able to continue to work with my favourite products Delphi, C ++ Builder and InterBase.
It was also a very exciting time. We had to do everything by ourselves now.
Run the office, process the orders, stock control, organise events, etc. And that is what I have been doing for more than 12 years now.
Our team grew and we moved to Haarlem.
We also have an office in France
(Barnsten is responsible for the sales and support activities for Embarcadero in the Benelux, France and the French speaking African countries).
We regularly organize events such as the annual Delphi Conference and we do the "Delphi Tour de France" in 6 to 10 cities in France.
We also organise many Meetups in collaboration with Delphi customers and produce webinars in cooperation with the MVP's we work with.

How do you see the future?
The future looks very promising.
Delphi is still widely used for the development of Windows applications because it is easy to learn and it is the best development environment for Windows.
We see the demand for developing Windows applications growing heavily again!
Delphi started to add a new path a few years ago with the support of FMX for cross-platform applications.
About 25% of our customers is now using this cross platform functionality to build also apps for iOS and Android often in combination with their Windows applications.
Delphi continues to innovate and grow with the needs of the market.
More than 2500 students in the Benelux are using the free community edition of Delphi at the moment.

Barnsten also setup contracts with 7 universities in The Netherlands for more than 4000 licenses.

In France we can double these figures (also because of the size of the country).
We also have set up a contest for the Best young developer in France and in the Benelux countries to get more awareness.
Delphi became inseparable in my life. I am so enthusiastic about all the companies and developers I have worked with.
I have seen innumerable applications and apps that are build in Delphi.
It helps businesses every day to do an excellent job . That gives me a all the energy I need to work with this great product every day.
I look forward to the next 25!



INTERNATIONAL PASCON EVENT 2020

DELPHI & LAZARUS

BLAISE PASCAL MAGAZINE

DELPHI & LAZARUS PASCON EVENT

Tuesday 22 (Delphi) and Wednesday 23 (Lazarus) April 2020

DELPHI KEYNOTE & ROAD MAP / WEBCORE / RESTFRAMEWORK FMXER / ANDREA MAGNI / KIM MADSEN / KBMMW ENTERPRISE WEBASSEMBLY / PAS2JS & GENERICS / FPC 3.2 / LAZARUS 2.2 LAZARUS HANDBOOK PRESENTATION & SIGNATURE SESSION

The event address:
"DE KLUIS" - Dr. Holtroplaan 1 Eindhoven / CONTACT: Mobile: +31 6 21.23.62.68
admin @ blaisepascalmagazine.eu / All English spoken. (German and Dutch translation available)


tmssoftware.com barnsten The Lazarus Factory COMPONENTS 4 DEVELOPERS

starter expert DX

INTRODUCTION:

In this special edition I want to explain how easy it is to load a ClientDataSet from CDS or file and then create a file for kbmMemTable as Binary or as Comma Separated value file: CSV.
This opens the possibility to collect your Client Dataset files and make use of the fantastic SQL method kbmMemTable offers. The end of filtering:
Now you can do SQL requests which is a far better way of collecting data. It is cleaner, quicker and definitively much more exact.
All the other functions of adding and changing your Dataset here is of course still available.
If you have any special request let me know.

For the functionality you need to consult some of the earlier publications 18/62/77/80/83. The lower number are of course quite old. So for the latest details you can use the higher numbers. (It is all Documented including code in the Library – The LIB – STICK or usb stick you can order)
https://www.blaisepascalmagazine.eu/product-category/lib-stick/

Now you can start the tool in two basic directions: You can either create a new Dataset or Load (import) one and then do what you normally cant: alter your existing fields, add fields even when your Dataset contains information. In this case we begin by loading a dataset. (see figures 1 and 2)

Figure 1: Load an existing dataset.

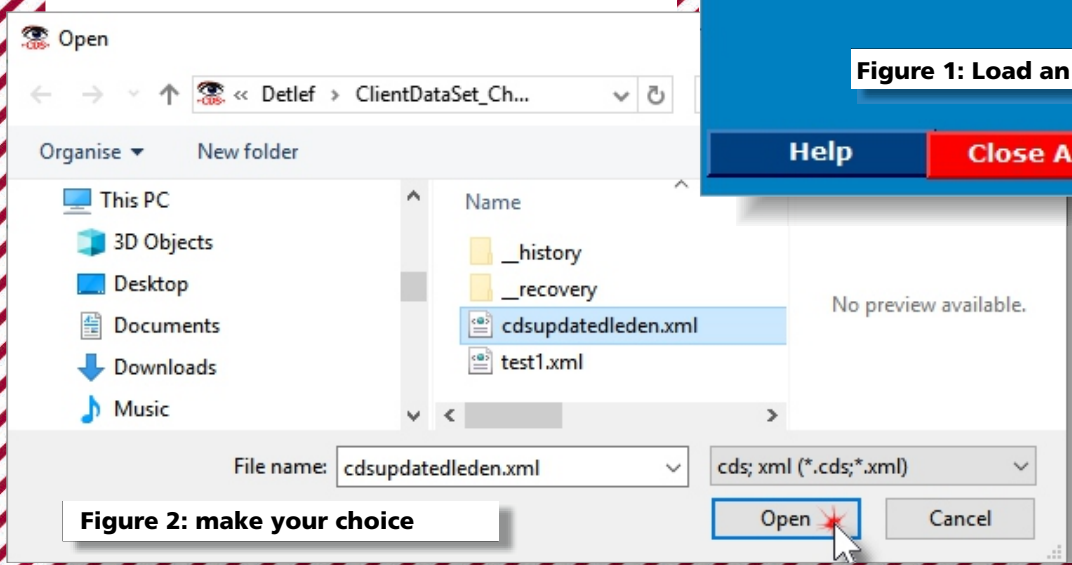


Figure 2: make your choice

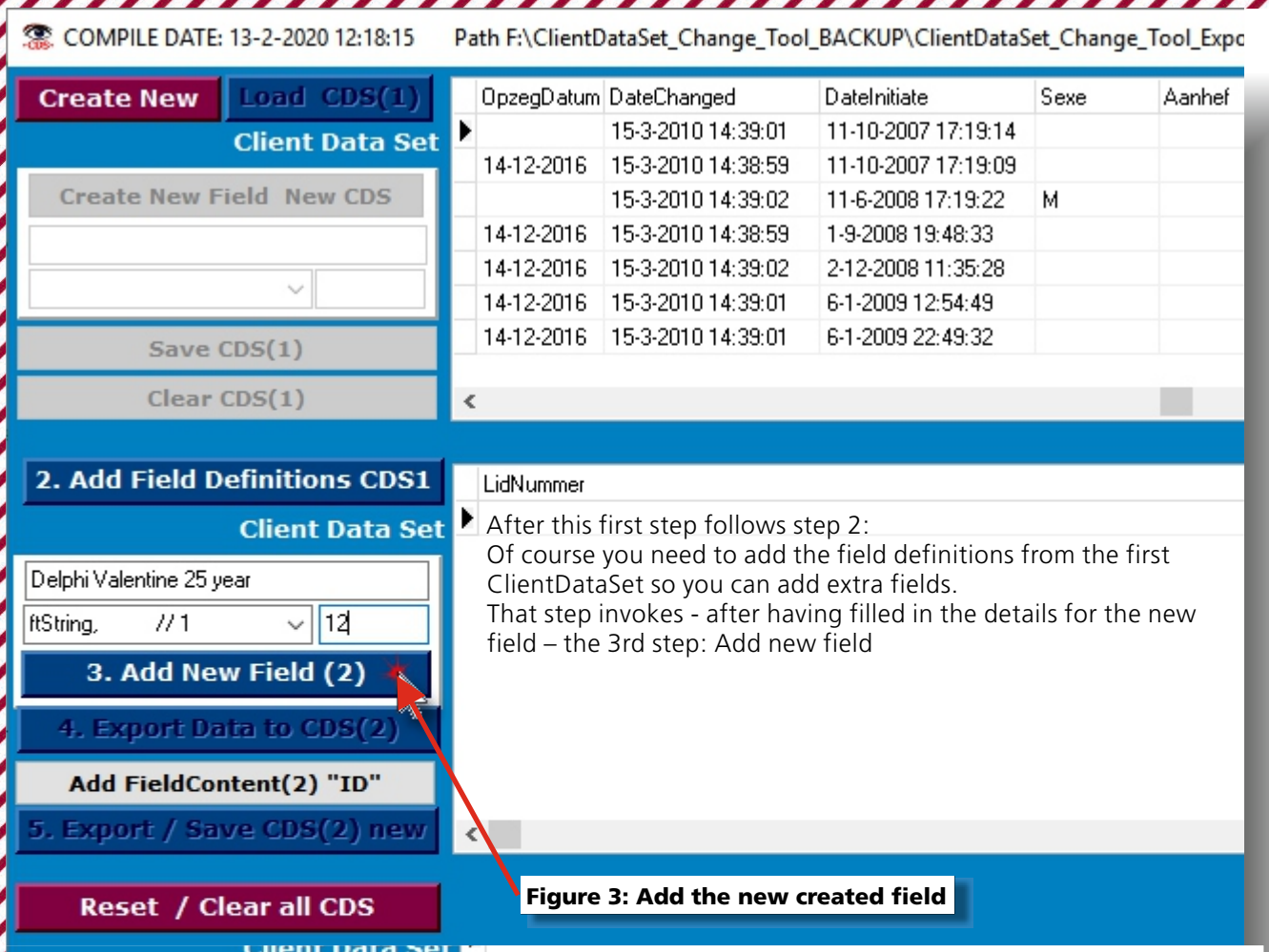


Figure 3: Add the new created field

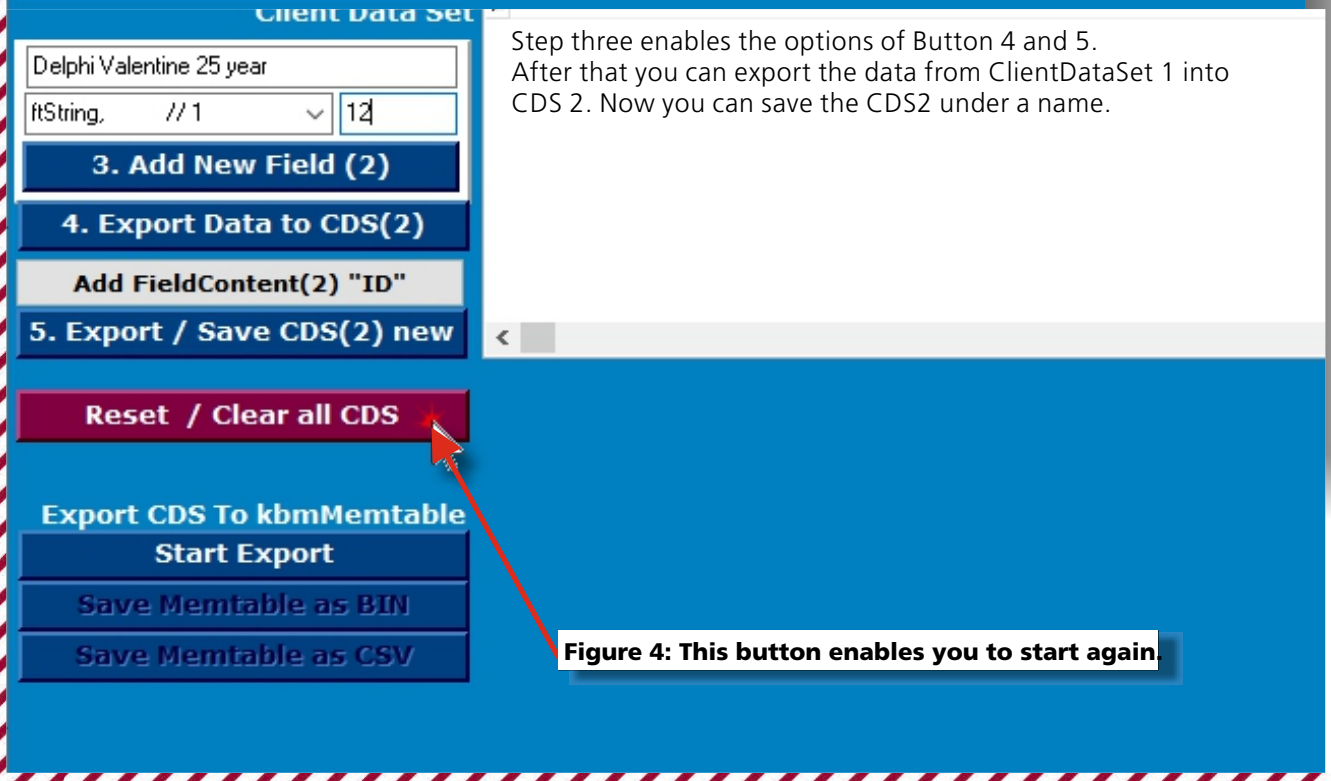
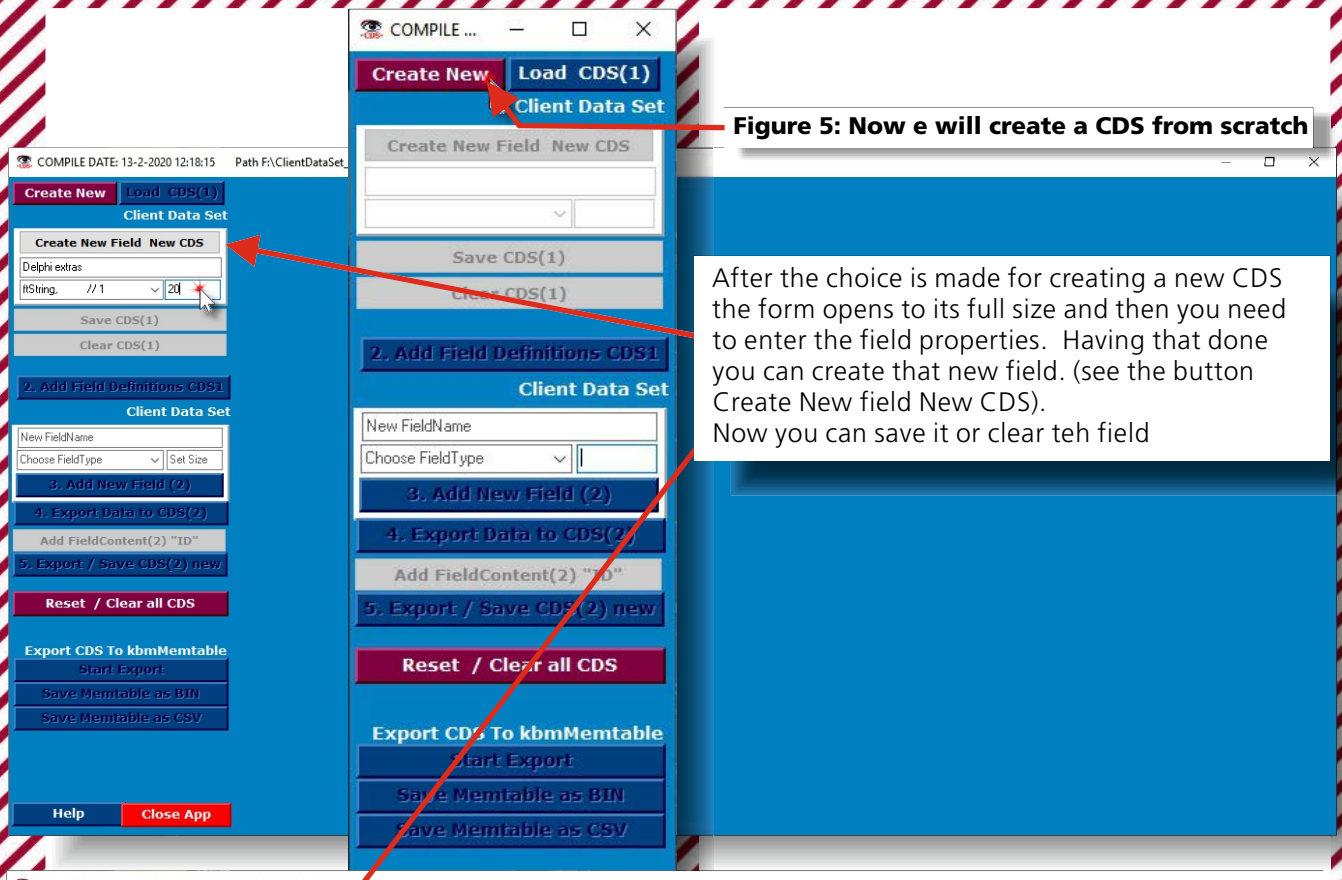


Figure 4: This button enables you to start again.


COMPILE ...
Create New
Load CDS(1)
Client Data Set
Create New Field New CDS
Save CDS(1)
2. Add Field Definitions CDS1
Client Data Set
New FieldName
Choose FieldType
3. Add New Field (2)
4. Export Data to CDS(2)
Add FieldContent(2) "ID"
5. Export / Save CDS(2) new
Reset / Clear all CDS
Export CDS To kbmMemtable
Start Export
Save Memtable as BIN
Save Memtable as CSV
Help
Close App
Figure 5: Now e will create a CDS from scratch
After the choice is made for creating a new CDS the form opens to its full size and then you need to enter the field properties. Having that done you can create that new field. (see the button Create New field New CDS).
Now you can save it or clear teh field

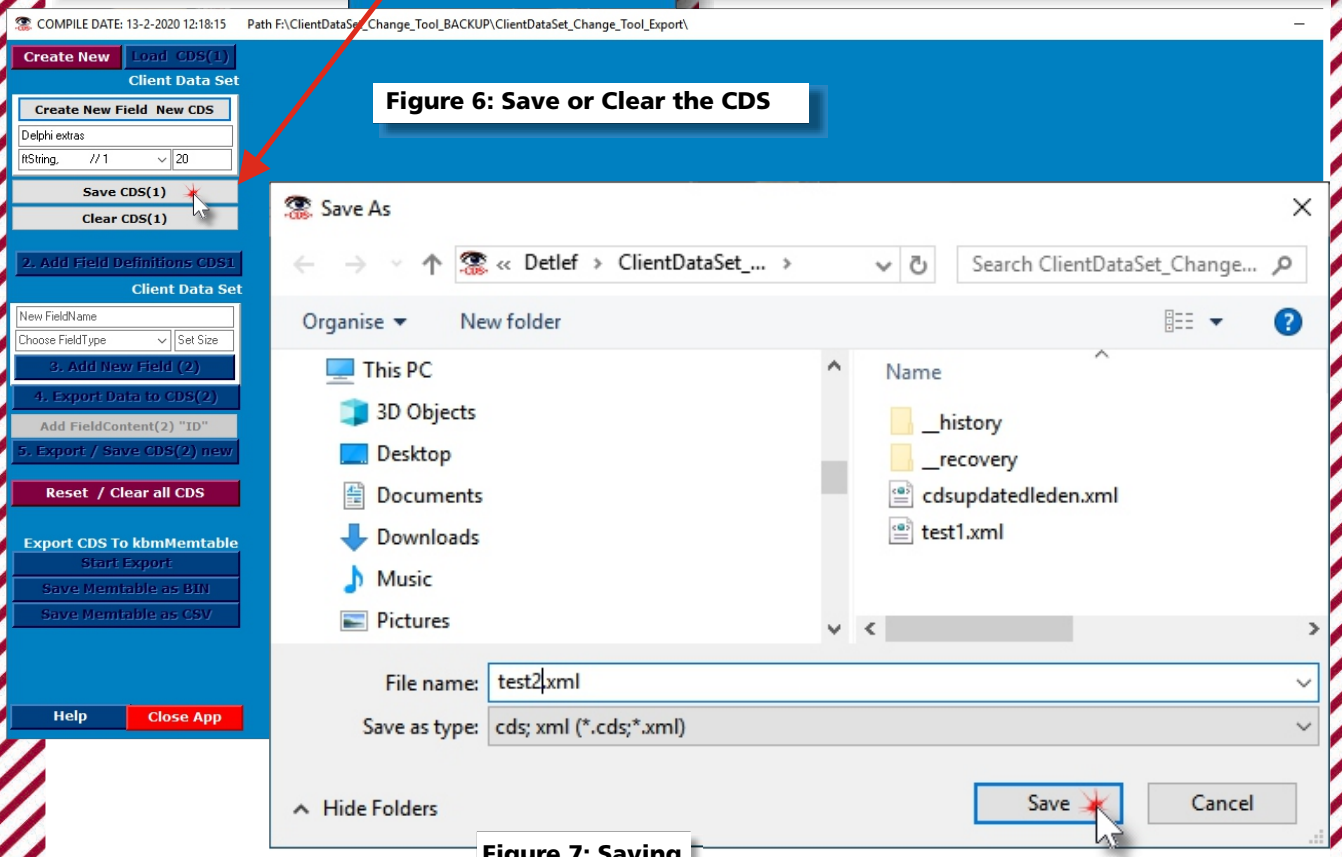


Figure 6: Save or Clear the CDS
Save As
Detlef
ClientDataSet_...
Search ClientDataSet_Change...
Organise
New folder
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Name
_history
_recovery
cdsupdatedleden.xml
test1.xml
File name: test2.xml
Save as type: cds; xml (*.cds;*.xml)
Hide Folders
Save
Cancel
Figure 7: Saving

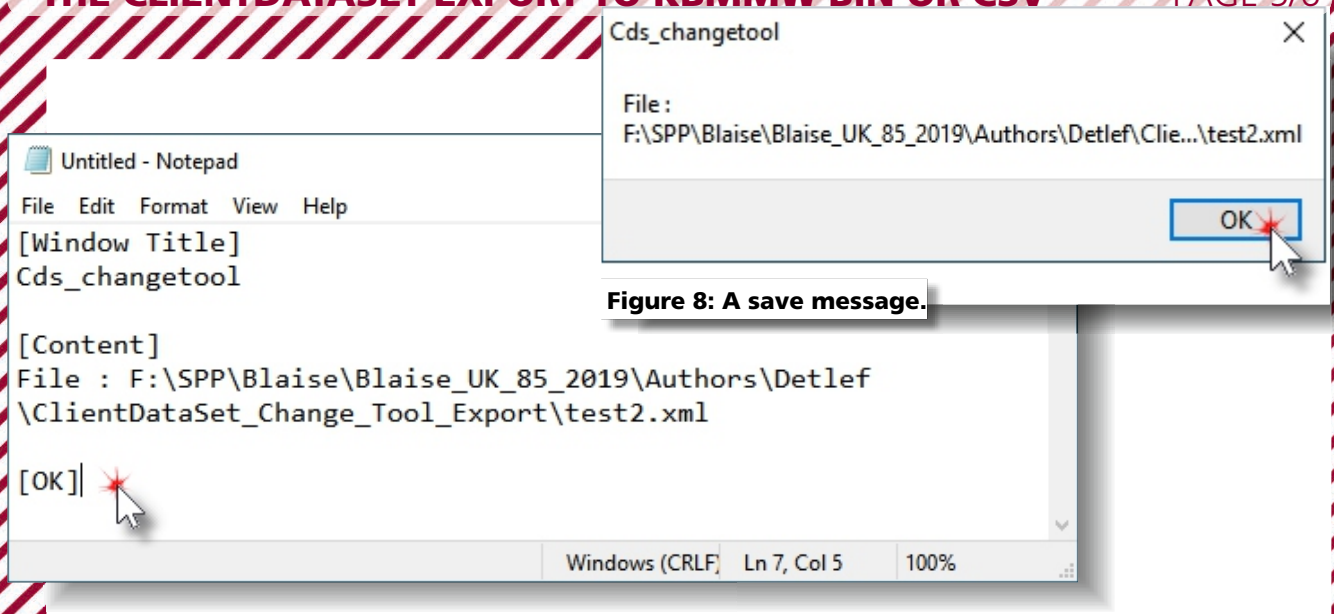
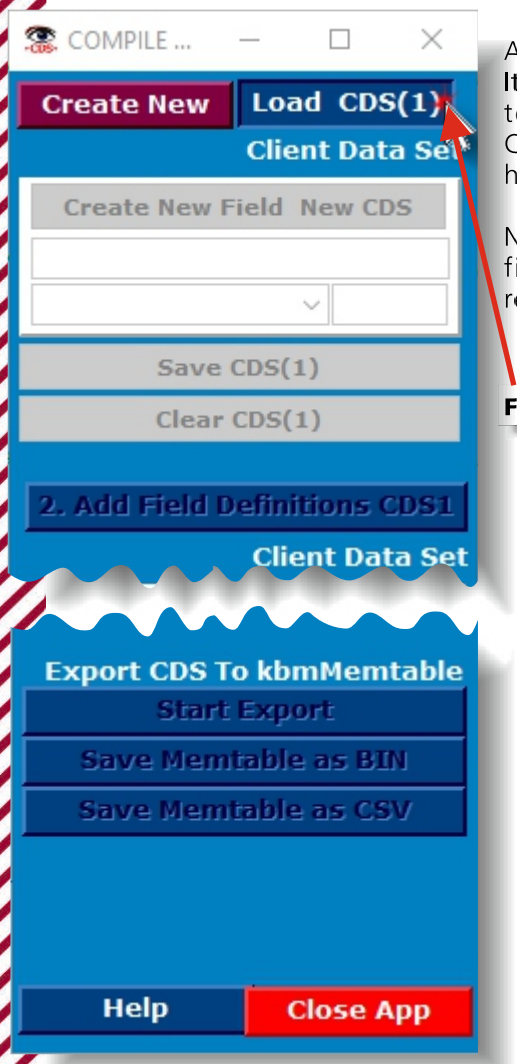


Figure 8: A save message.



After you saved the CDS a new window ill pop up: See figure.
It might be good to know that under windows you can recreate the text op the message by entering
Ctrl C. In that way you can save the messages for later use. (Could have been an error message).

Now I want to finally export the complete ClientDataSet with the field content to the kbmMemTable. So then I could finally use SQL requests on that table. The app had shrunk to its minimum size

Figure 9: Click for opening and export

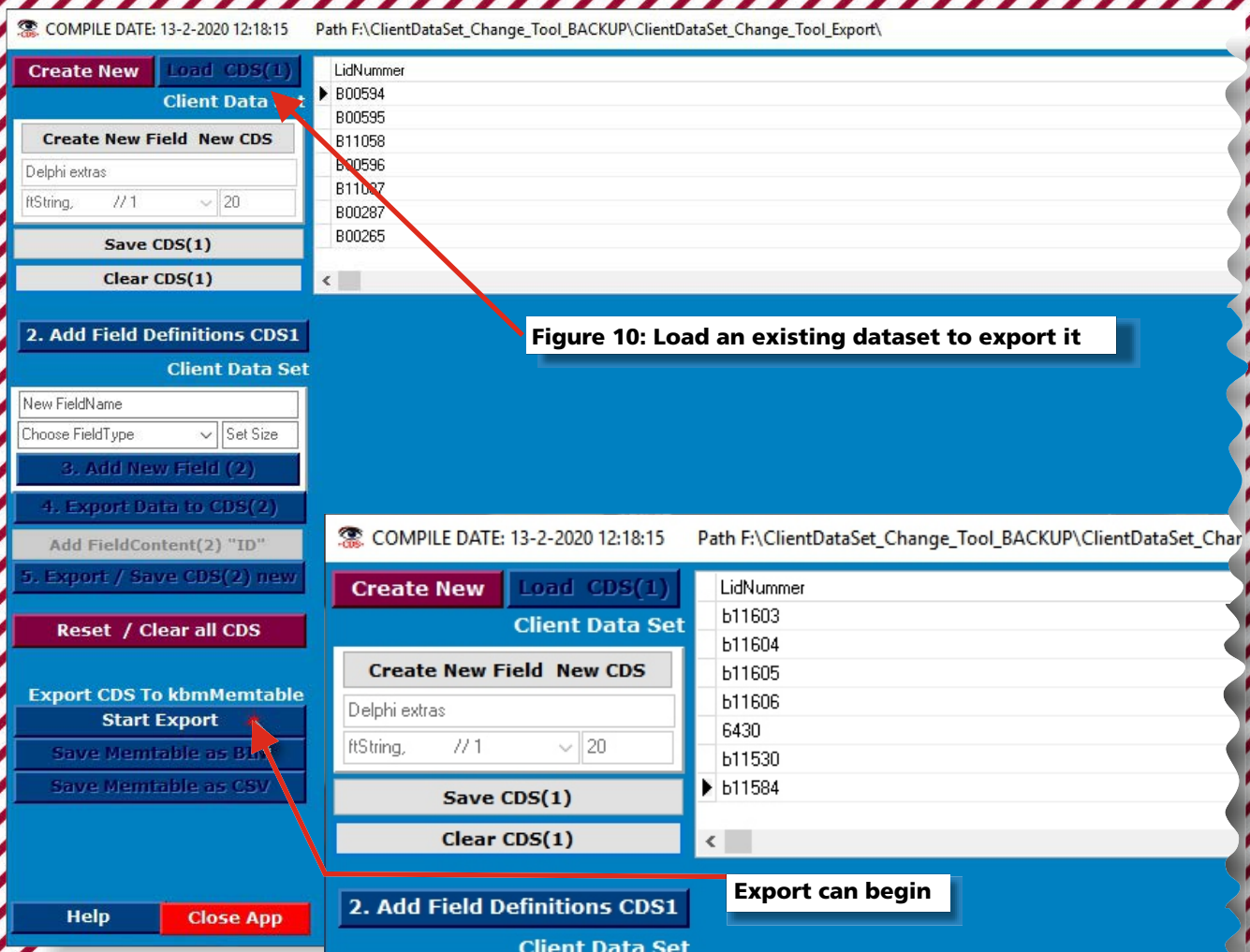


Figure 10: Load an existing dataset to export it

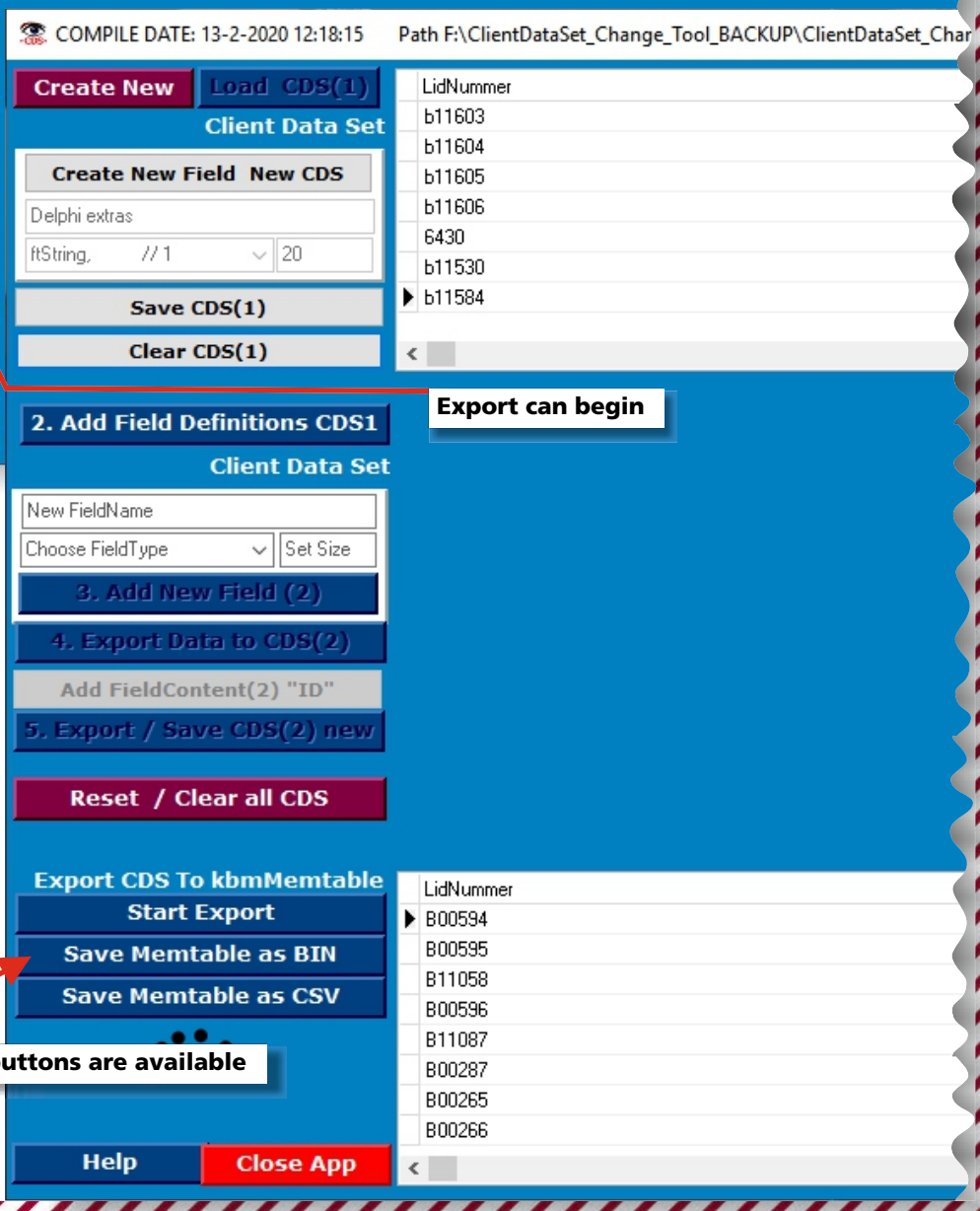


Figure 11: All buttons are available

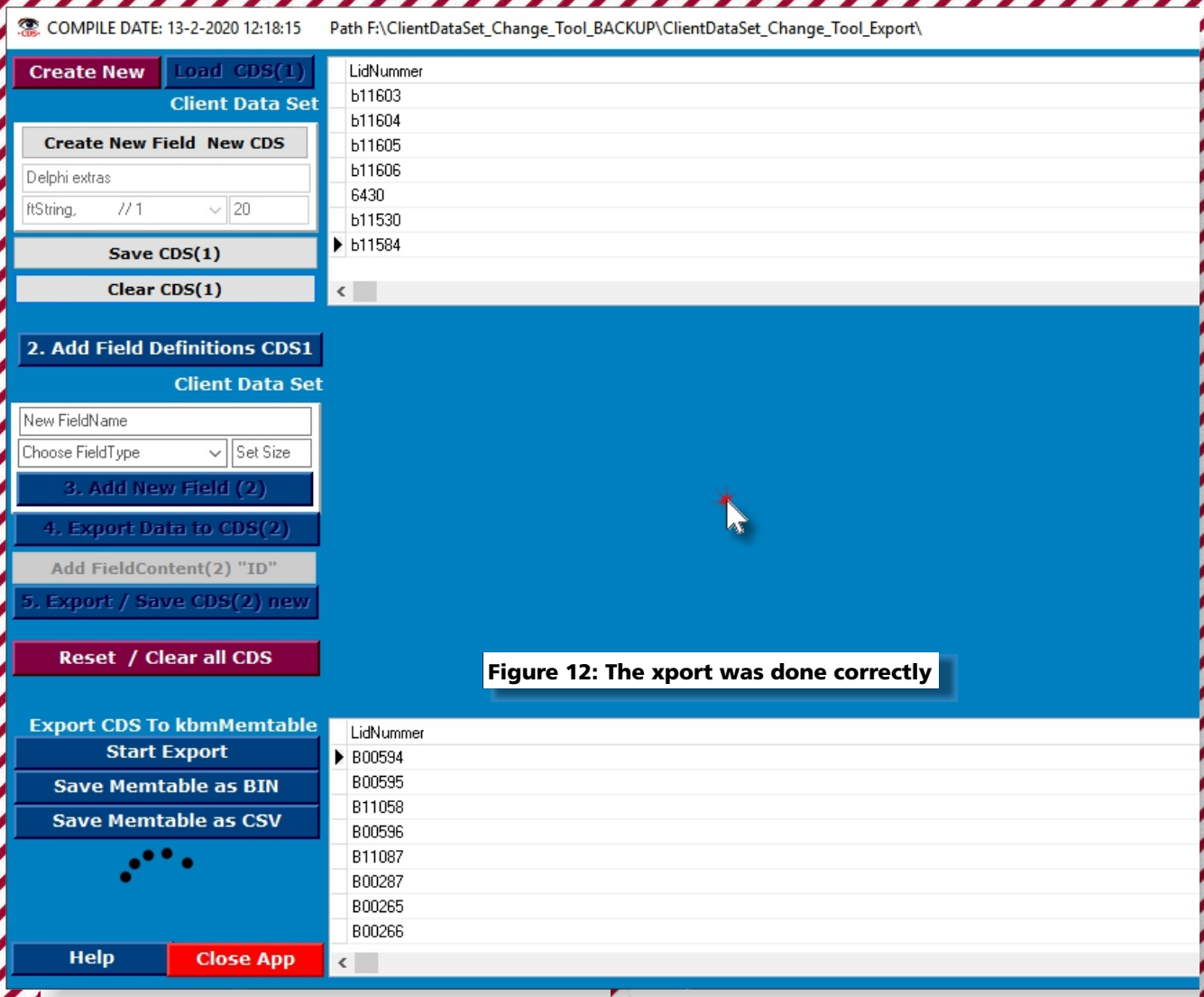


Figure 12: The xport was done correctly

Figure 13: Save as BIN

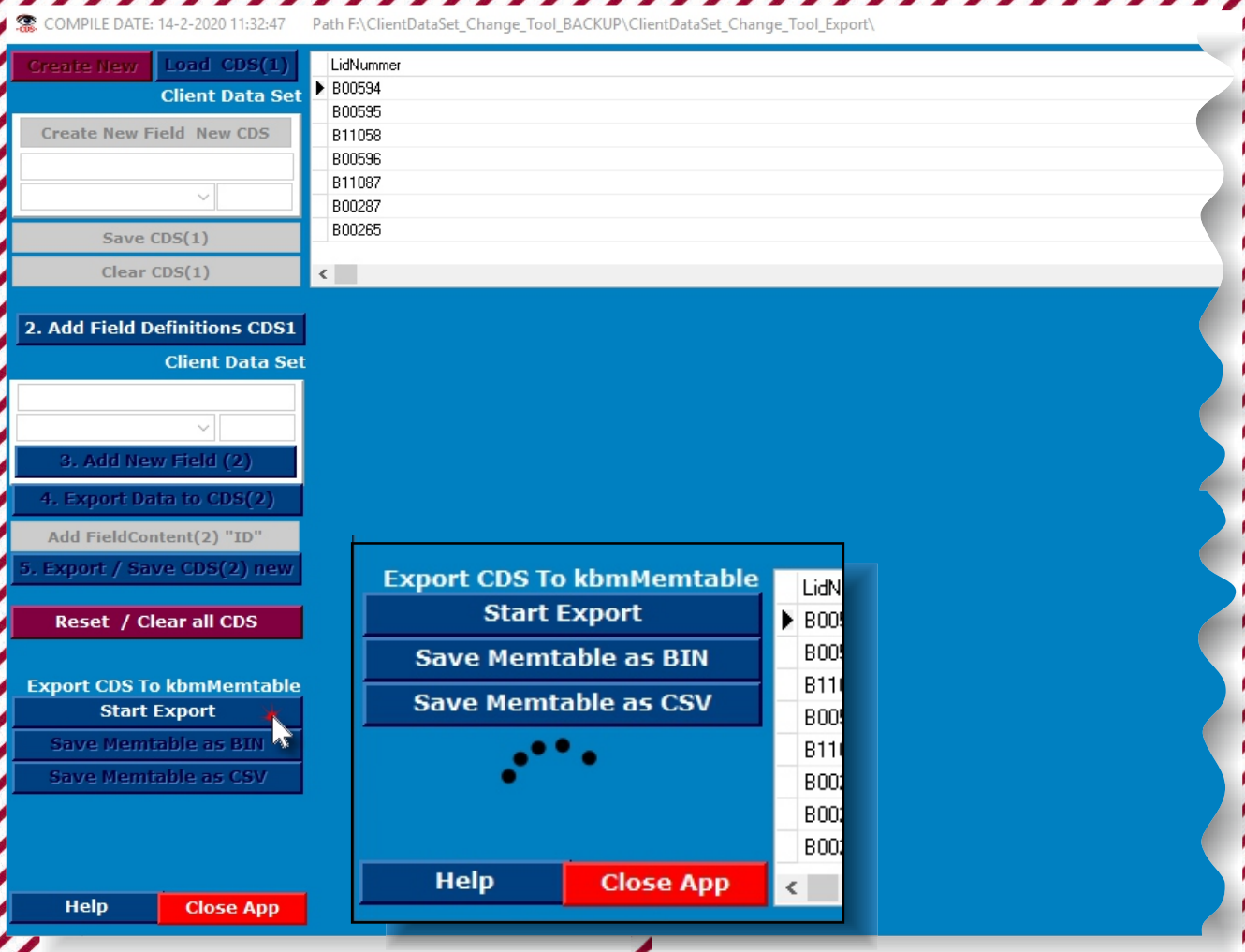
Figure 14: Save as Comma Separated Value

I show here the implementation that has to be done in our code if you want to use the **TActivityIndicator:**
It will be triggered by pushing the Start Export Button. First of all one needs to create a thread:

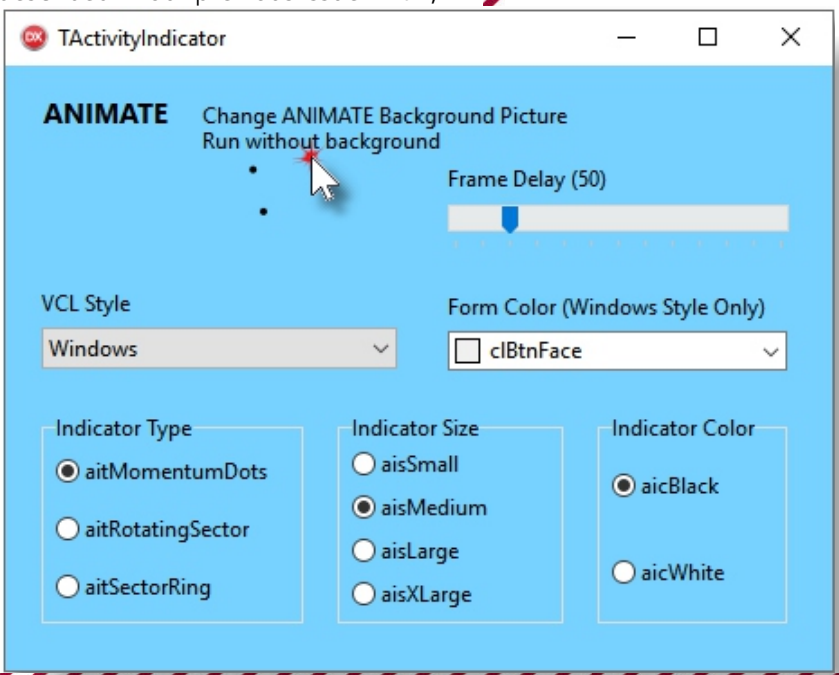
```
Implementation

{$R *.dfm}
////////////////////////////////////// Form
////////////////////////////////////

type
 TExportThread = class(TThread)
 protected
  procedure Execute; override;
 public
  kbmMemTable1: TkbmMemTable;
  CDS1: TClientDataSet;
  Finished: boolean;
 end;
```

After this all a number of things have to be started and ended. The code is easy to understand

```
 procedure TGUI.ColorButton1Click(Sender: TObject);
Var    I : Integer;
     ExportThread: TExportThread;
Begin
 ClrBtnSaveMemtableBIN.Enabled := True;
 ClrBtnSaveCSV.Enabled := True;

 DBGrid1.Visible := False; // the data source is disengaged to make the
 DBGrid1.DataSource:=nil; // load time event as small as possible
 DBGrid2.Visible := False;
 DBGrid3.Visible := False;
 DBGrid3.DataSource:=nil;

 ActivityIndicator1.Animate := True;

 ExportThread:=TExportThread.Create(true);
 try
  ExportThread.FreeOnTerminate:=false;
  ExportThread.kbmMemTable1:=kbmMemTable1;
  ExportThread.CDS1:=CDS1;
  ExportThread.Start;
  repeat
   Sleep(10); // let  the main thread sleep shortly
   Application.ProcessMessages; // triggers timer of ActivityIndicator1
  until ExportThread.Finished;
 finally
  ExportThread.Free;
 end;

 DBGrid1.Visible := True;
 DBGrid1.DataSource:=DS1;
 DBGrid3.Visible := True;
 DBGrid3.DataSource:=DS1;

 ActivityIndicator1.Animate := False;
end;
```



The **`ActivityIndicator Component`** was extensively described in our previous issue nr 84, at page 39.



tmssoftware.com

PREREQUISITES VIDEOS FAQ PRODUCTS REGISTER

develop • faster

TMS Academic License
TMS VCL UI Pack Academic License

Designed for Delphi Community Edition

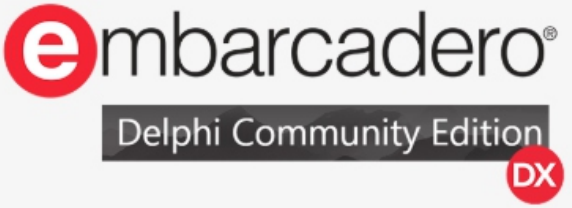
TMS Academic License

Getting Started

Prerequisites

Designed for Delphi Community Edition

- TMS Academic licensed products work with Delphi Community Edition
  Discover the latest free Delphi Community Edition from Embarcadero
- Start developing applications for Windows and cross-platform
- Build powerful database powered applications with hundreds of components
- Visual design using VCL or FMX and debug from the IDE



TMS Academic License

- Fully functional and fully free versions of TMS Delphi products for students for non-commercial use
- Full experience including account on our website and access to our forums
- Free version can be used with the latest Delphi Community Edition
- First academic version is for TMS VCL UI Pack, offering over 600 VCL UI controls for your Windows applications
- More academic versions of products will be added in the coming months

Getting Started

- Register below for a free TMS VCL UI Pack Academic license with a school or university email address
- Credentials to login on our website will be send to your student email address
- Login and download your free TMS VCL UI Pack Academic installer
- Install for the Delphi Community Edition
- Need help? Watch our videos, read the manual or contact our team

1. Register
2. Install
3. Develop

```
<?xml version="1.0"?>
<quiz>
 <qanda seq="1">
  <question>
   Who was the forty-second
   president of the U.S.A.?
  </question>
  <answer>
   William Jefferson Clinton
  </answer>
 </qanda>
 <!-- Note: We need to add
  more questions later.-->
</quiz>
```

XML

HOW TO PARSE XML IN 3 DIFFERENT WAYS
using kbmMW Pro, Enterprise and Community Edition.
This article is based on a public question on the NLDelphi.com forums.

Having the following XML, how do can it be parsed?

```
const
  XML: string =
   '<?xml version="1.0"?>'+
   '<DefaultBody>'+
   '<Default>'+
   '  <Defaultcode>XML_BOLIMPORT_MAP</Defaultcode>'+
   '  <Omschrijving>Folder waar de xml bestanden staan,
       die gemaakt zijn door bolmate vanuit de
       inkoop</Omschrijving>'+
   '  <Waarde>Z:\tmp</Waarde>'+
   '</Default>'+
   '<Default>'+
   '  <Defaultcode>XML_DESTUSED_MAP</Defaultcode>'+
   '  <Omschrijving>XMLBestanden die verplaatst worden
              als ze klaar zijn</Omschrijving>'+
   '  <Waarde>Z:\tmp\xmlexport</Waarde>'+
   '</Default>'+
```

Into a list of TDef objects:

```
type
 TDef = class
 public
   Defaultcode:string;
   Omschrijving:string;
   Waarde:string;
 end;

 TDefs = class(TObjectList<TDef>);
```

```
using
 kbmMWXML;
...
function ParseXML1(const AString:string):TDefs;
var xml:TkbmMWDOMXML; i:integer;
  data,node,child:TkbmMWDOMXMLNode; def:TDef;
begin
   Result:=TDefs.Create;
   xml:=TkbmMWDOMXML.Create(AString);
   try
     data:=xml.Top; // Will usually be DefaultBody node
     for i:=0 to data.Nodes.Count-1 do
     begin
        node:=data.Nodes[i];
        if node.Name='Default' then
        begin
           def:=TDef.Create;
           child:=node.ChildByName['Defaultcode'];
           if child<>nil then
             def.DefaultCode:=child.GetAsString;
           child:=node.ChildByName['Omschrijving'];
           if child<>nil then
             def.Omschrijving:=child.GetAsString;
           child:=node.ChildByName['Waarde'];
           if child<>nil then
             def.Waarde:=child.GetAsString;
           Result.Add(def);
        end;
     end;
   finally
     xml.Free;
   end;
end;
```

Method 2 – Using object notation
Notice there are many more dynamic ways to access the data using object notations than just this example. For code see next page...

Method 2 – Using object notation

```
uses
 kbmMWXML,
 kbmMWObjectNotation;

...

function ParseXML2(const AString:string):TDefs;
var
  xml:TkbmMWDOMXML;
  co:TkbmMWONCustomObject;
  o:TkbmMWONObject;
  def:TDef;
begin
   Result:=TDefs.Create;
   xml:=TkbmMWDOMXML.Create(AString);
   try
     def:=TDef.Create;
     co:=xml.SaveToObjectNotation;

     def.DefaultCode:=TkbmMWONNative(co.Path('DefaultBody/Default/0/Defaultcode',[mwonpoLastIsValue],
           '/',mwontString)).AsString;
     def.Omschrijving:=TkbmMWONNative(co.Path('DefaultBody/Default/0/Omschrijving',[mwonpoLastIsValue],
           '/',mwontString)).AsString;
     def.Waarde:=TkbmMWONNative(co.Path('DefaultBody/Default/0/Waarde',[mwonpoLastIsValue],
           '/',mwontString)).AsString;
     Result.Add(def);

     def:=TDef.Create;
     def.DefaultCode:=TkbmMWONNative(co.Path('DefaultBody/Default/1/Defaultcode',[mwonpoLastIsValue],
           '/',mwontString)).AsString;
     def.Omschrijving:=TkbmMWONNative(co.Path('DefaultBody/Default/1/Omschrijving',[mwonpoLastIsValue],
           '/',mwontString)).AsString;
     def.Waarde:=TkbmMWONNative(co.Path('DefaultBody/Default/1/Waarde',[mwonpoLastIsValue],
           '/',mwontString)).AsString;
     Result.Add(def);


   finally
     co.Free;
     xml.Free;
   end;
end;
```

Method 2 – Using object notation
Notice there are many more dynamic ways to access the data using object notations than just this example. For code see next page...

Method 3 – Using object marshalling

This is by far the easiest way to parse the data, and also makes it very easy to generate XML from the objects again.

Notice that the **TDef** and **TDefs** class definitions has been augmented with a couple of attributes.

```
uses
 Generics.Collections,
 kbmMWObjectMarshal,
 kbmMWXMLMarshal,
 kbmMWRTTI;

type

  [kbmMW_Root('Default',[mwrfIncludePublic])]
  TDef = class
  public
    Defaultcode:string;
    Omschrijving:string;
    Waarde:string;
  end;

  [kbmMW_Root('DefaultBody')]
  [kbmMW_Child('Default')]
  TDefs = class(TObjectList<TDef>);

...

function ParseXML3(const AString:string):TDefs;
var
  xml:TkbmMWXMLMarshal;
begin
   xml:=TkbmMWXMLMarshal.Create;
   try
     Result:=xml.FromString<TDefs>(AString);
   finally
     xml.Free;
   end;
end;

...

initialization
  kbmMWRegisterKnownClasses([TDef,TDefs]);
```

Method 2 – Using object notation

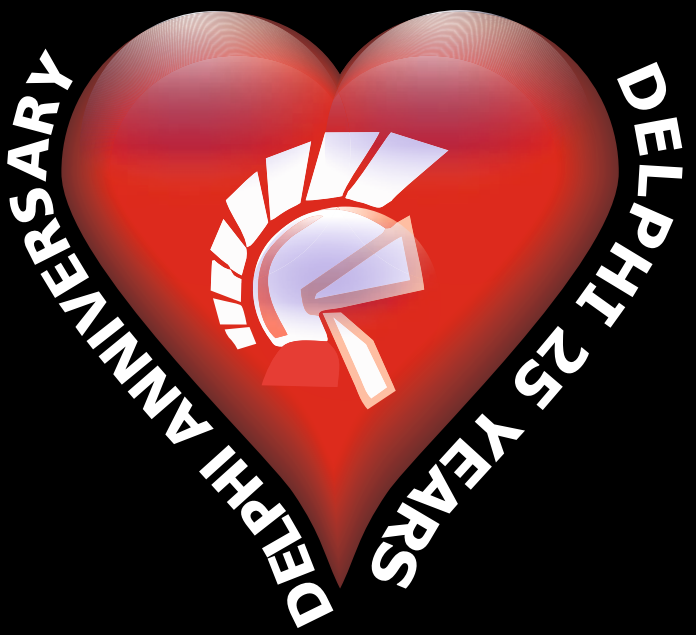
Notice there are many more dynamic ways to access the data using object notations than just this example. For code see next page...

THE KBMMW COMMUNITY EDITION HAS BEEN RELEASED

FOR FREE

to celebrate 25 years of DELPHI

kbmMW Community Edition contains both
kbmMW Enterprise like features and
kbmMemTable Standard features



COMPONENTS 4 DEVELOPERS

DX

Want to find energy at night?

KBMMW PROFESSIONAL AND ENTERPRISE EDITION V. 5.10.20 RELEASED!

- RAD Studio XE2 to 10.3 Rio supported
- Win32, Win64, Linux64, Android, IOS 32, IOS 64 and OSX client and server support
- Native high performance 100% developer defined application server
- Full support for centralized and distributed load balancing and failover
- Advanced ORM/OPF support including support of existing databases
- Advanced logging support
- Advanced configuration framework
- Advanced scheduling support for easy access to multithread programming
- Advanced smart service and clients for very easy publication of functionality
- High quality random functions.
- High quality pronouncable password generators.
- High performance LZ4 and Jpeg compression
- Complete object notation framework including full support for YAML, BSON, Messagepack, JSON and XML
- Advanced object and value marshalling to and from YAML, BSON, Messagepack, JSON and XML
- High performance native TCP transport support
- High performance HTTPSys transport for Windows.
- CORS support in REST/HTML services.
- Native PHP, Java, OCX, ANSI C, C#, Apache Flex client support!

kbmMemTable is the fastest and most feature rich in memory table for Embarcadero products.

- **Easily supports large datasets with millions of records**
- **Easy data streaming support**
- **Optional to use native SQL engine**
- **Supports nested transactions and undo**
- **Native and fast build in M/D, aggregation/grouping, range selection features**
- **Advanced indexing features for extreme performance**

- **NEW! SmartBind now fully supports VCL, FMX, including image/graphics and TListView**
- **NEW! SmartBind data generators and data proxies for easy separation of data sharing concerns in modular applications**
- **NEW! SmartEvent for easy separation of event and execution workflow based concerns for the ultimate in modular application design**
- **NEW! Native highly scalable TCP server transport now also supports REST**
- **Significant improvements and fixes in many areas including**
  - **RTTI**
  - **Scheduler**
  - **LINQ**
  - **Object Notation**
  - **ORM**

- High speed, unified database access (35+ supported database APIs) with connection pooling, metadata and data caching on all tiers
- Multi head access to the application server, via REST/AJAX, native binary, Publish/Subscribe, SOAP, XML, RTMP from web browsers, embedded devices, linked application servers, PCs, mobile devices, Java systems and many more clients
- Complete support for hosting FastCGI based applications (PHP/Ruby/Perl/Python typically)
- Native complete AMQP 0.91 support (Advanced Message Queuing Protocol)
- Complete end 2 end secure brandable Remote Desktop with near realtime HD video, 8 monitor support, texture detection, compression and clipboard sharing.
- Bundling kbmMemTable Professional which is the fastest and most feature rich in memory table for Embarcadero products.

COMPONENTS 4 DEVELOPERS

EESB, SOA,MoM, EAI TOOLS FOR INTELLIGENT SOLUTIONS. kbmMW IS THE PREMIERE N-TIER PRODUCT FOR DELPHI / C++BUILDER