

ProIV IDE (Integrated Development Environment)

Writer: [Rob Donovan](#)

Company: [Parallax IT LTD](#)

Published: [March 2011](#)

Summary: This document discusses the advantages of using ProIV IDE in a development, analysis and production support environment.

Overview

ProIV has never had a solid well designed development environment, and has been seriously underdeveloped for many years. Many coders and sites over time have therefore been obliged to write their own utilities to help with more efficient development.

Several attempts by the owners of ProIV have tried to address some of these problems, but they have failed to integrate and unify the development process.

To some extent they have actually made it harder to code and visualize the ProIV flow of code.

It was clear that the ProIV community needed a method to group these tools into a single development environment which would be designed and laid out in an intuitive and visually pleasing way. This need gave birth to ProIV IDE.

Over many years the developer of ProIV IDE consulted with ProIV coders around the world to try to find the best way to present a ProIV function and integrate tools into the environment the best way to make development not only more fun, but also more productive.

ProIV IDE is the best way to make development not only more fun, but also more productive and it speaks to developers in a way that makes sense.

Many of the ideas and features in ProIV IDE have been directly requested or inspired by some of the world's top ProIV specialists.

It's not just a development tool; it's a whole environment for ProIV. It greatly helps in the process of Analysis, system design and IS/DS writing.

Also, with ProSQL and other utilities in ProIV IDE, production support and error finding tasks can be hugely reduced and simplified.

Some of the world's top ProIV coders have said that ProIV IDE can easily reduce workload by 25-30% when using ProIV IDE over other ProIV tools.

This has a positive and desired effect on the company's bottom line, enabling developers to deliver solutions to the market quicker, with the highest quality achievable compared to any other ProIV development environment.

So, what does ProIV IDE give you?

The following paper has been written to explain how ProIV IDE works. Described herein are the features and tools available that make ProIV IDE the best choice of environments. Thereby allowing a development team to excel.

History

ProIV IDE development started out some 12 years ago. The first version never really got out of development, due to problems with trying to communicate with ProIV.

After a few years of little development, the main development work started again in 2001.

After many years of work and testing, it started to get used at a site in Australia and then in 2006 in Sweden.

In 2009 the source code was moved to the offices of a small UK IT company called Parallax IT LTD.

ProIV IDE is continually developed and maintained, and new features and ideas are added, to continue to make it the fastest development tool for ProIV.

About the developer

ProIV IDE is the brainchild of long term ProIV developer, Rob Donovan who has worked in ProIV since 1989, working at various sites around the world.

His main focus has been on performance and utilities for the language, but has also been involved with lots of development for many applications.

Not only coding in ProIV, he has experience in VB, C, PHP & web development. He is also experienced in Unix / Linux and the Windows OS, and also file/databases including Oracle, MySQL & C-ISAM.

He's also helped debug and improve performance on the Linux kernel, mainly to help with C-ISAM and ProIV, and also the author of www.proivrc.com, the ProIV Resource Center.

How it works

ProIV IDE is written in Visual Basic, incorporating some 3rd party controls to keep up with modern interfaces and design. It includes controls from Component One for grid control & Innovasys for the docking, windows and navigation UI. Both these companies have commercial products that are integrated in many applications.

ProIV source code is stored in a group of ProISAM files (known as the bootstrap files), only accessible from within the ProIV environment.

So there are 2 parts to ProIV IDE.

Client Side

The client side of ProIV IDE is the application that sits on the developers PC and what they use to 'edit' the ProIV code.

It needs to be installed on the user's PC, and it uses SSH connections to the server to communicate with ProIV.

Server Side

The server side consists of a controlling Unix script and ProIV functions that extract & import the ProIV source code to and from the client. It also performs other server side operations that are more suited to the server side for performance. All communication is passed via a flat file method in a specific format designed for ProIV IDE.

All transfers are zipped for compression and SSH is used to ensure security. All connections are using standard SSH login and password methods.

The Client and Server can be located within an internal corporate network or even across the internet and the connection is stateless, which allows ProIV IDE to recover from any network loss, seamlessly.

All changes made to the ProIV code are initially stored on the client side, and then sent to the server when the user decides to 'save' the changes.

ProIV code, once loaded from the server can also be saved client side, in text format for future loading and examination. Typically useful for if user knows they won't have a connection to the server and still wish to look at the ProIV code.



Features

Main Editing window

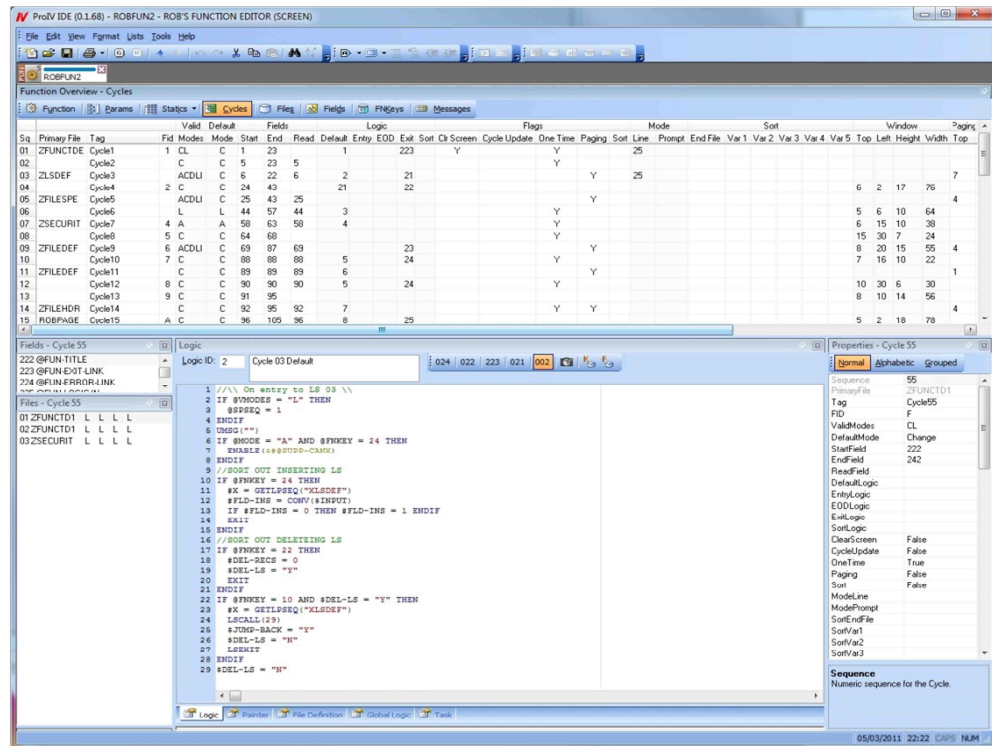
This is the main window in ProIV IDE. The user can get an immediate overview of the function here, and see the whole function in one view.

From a developer's perspective, this overview is one of the key strengths of ProIV IDE and allows the developer to take in the function at a glance and not have to drill down several menus to get a picture of how the function behaves.

Multiple Functions, Files, Global Logics & Tasks can be opened and easily navigated between in one easy to view place.

Developers can switch quickly between multiple functions, even copying chunks of code between different functions easily.

All windows and panes can easily be moved, resized and tabbed with standard windows docking techniques, and no modal windows are used that stop the user for being able to get to other windows.

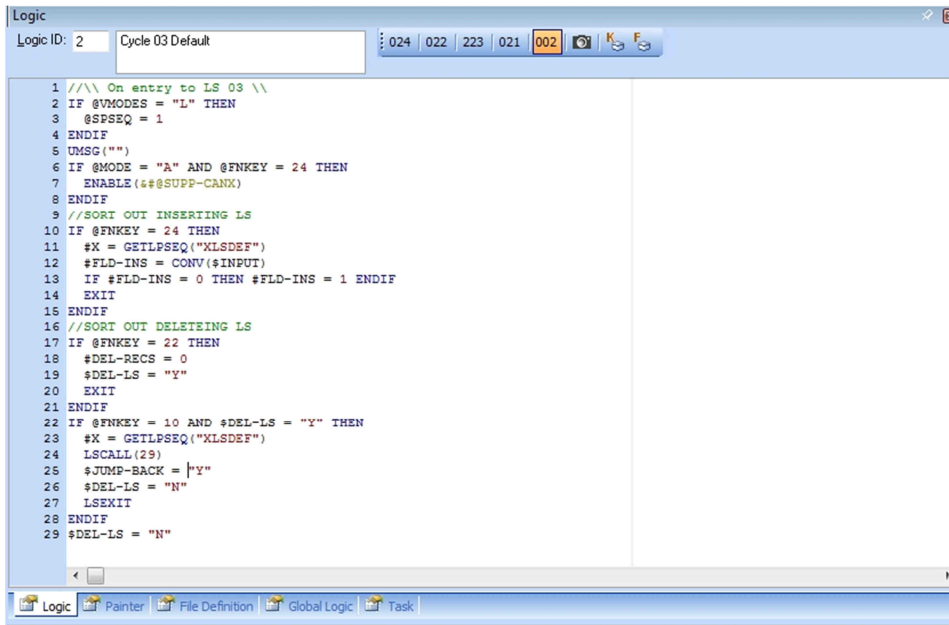


Once an item has been opened, all movement and modifications are done locally, so speed of navigation is very fluid and quick.

Everything the user needs to see to get a general overview of the function is displayed once the function has been opened. The developer can move, resize or close any of the windows if they wish. All window settings will be saved automatically.

Logic Editor

This is the window used to edit the actual ProIV 'code' part of a function. It's fully colour coded & formatted. A history bar at the top keeps track of recent logic IDs, and shows where the logic is called from. Right click menu, and toolbars also expose all utilities regarding logic.



```
1 //\ On entry to LS 03 \\
2 IF @VMODES = "L" THEN
3   @SPSEQ = 1
4 ENDIF
5 UMSG("")
6 IF @MODE = "A" AND @FNKEY = 24 THEN
7   ENABLE(##@SUPP-CANX)
8 ENDIF
9 //SORT OUT INSERTING LS
10 IF @FNKEY = 24 THEN
11   #X = GEILPSEQ("XLSDEF")
12   #FLD-INS = CONV($INPUT)
13   IF #FLD-INS = 0 THEN #FLD-INS = 1 ENDIF
14   EXIT
15 ENDIF
16 //SORT OUT DELETING LS
17 IF @FNKEY = 22 THEN
18   #DEL-RECS = 0
19   $DEL-LS = "Y"
20   EXIT
21 ENDIF
22 IF @FNKEY = 10 AND $DEL-LS = "Y" THEN
23   #X = GEILPSEQ("XLSDEF")
24   LSCALL(29)
25   $JUMP-BACK = "Y"
26   $DEL-LS = "N"
27   LSEXIT
28 ENDIF
29 $DEL-LS = "N"
```

Some logic utilities are:

- Copy, Cut & Paste
- Comment blocks
- Indent blocks
- Print
- Auto-Indent
- Logic snapshot
- Auto get of file fields
- Bookmarks

Other features like Compile error highlighting, double click loading for global functions, Auto Logic ID generation, and auto complete make editing logic very easy and quick.

As the developer types, syntax tooltips pop-up, giving them information on what the syntax of the current command is.

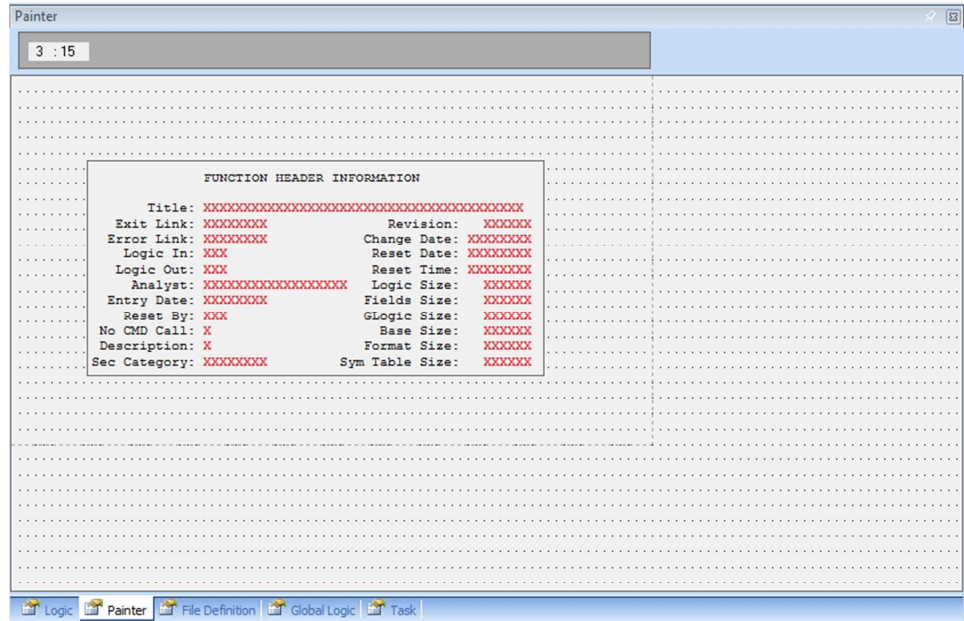
This further helps the developer by helping to eliminate possible errors in their code even before the function is compiled, once again adding to the speed with which they can develop.

Painter

The painter window allows the developer to view the screen designs in a graphical way, and allows them to drag & drop fields and layouts, making screen design / creation and modification quick and painless.

The painter clearly depicts screen boundaries, paging screen areas and windows, and treats them as objects that can be moved and positioned.

Highlighting of fields is used to show which items are 'editable' by the user, once the screen is running. Clicking on each item displays the full properties for that item in the 'Properties' window.



Tree View

This is a unique alternative design view that is based upon the ProIV timing cycle. It also displays the function in a more graphical and modern way, compared to the traditional way a function has been viewed.

	Modes	Fun		Cycle				Fields				File Read			File Write			File		Cycle		Fun
		Ent	Def	Srt	Bef	Gen	Aft	Dsp	Win	FNKeys	Bef	Err	Aft	Bef	Err	Aft	BefS	Lck	EOD	Exit	Exit	
ROBFUN2			96																		254	
Cycle: 01	C L			1																	223	
FNKeys																						
Files																						
Fields																						
Field: 001 - \$\$SECOND...																						
Field: 002 - \$\$FIRST-LINE																						
Field: 003 - \$\$FUNT																					147	
Field: 004 - \$FUNCTIO...																					95 98 97	
Cycle: 02	C																					
Fields																						
Cycle: 03	A C D L I			2																	21	
Files																						
File: 01 - ZLSDEF	A C D L																				171	
File: 02 - ZFILESPE	L L L L																				112 132 152	
Fields																						

All parts of the function are displayed here, and the user can get a full overview of the function easily, collapsing / hiding parts that aren't needed in their current development or analysis.

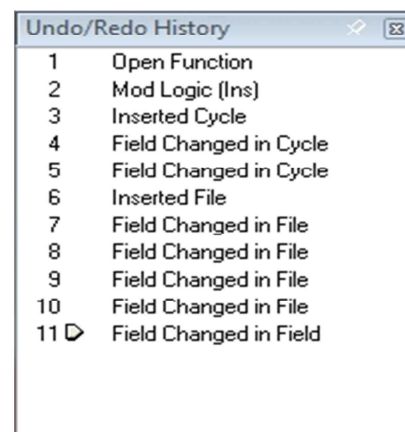
The function is displayed in a logical way, using the 'tree' feature to allow a visual way to display the ProIV timing cycle.

This can give a clear display of how a function will actually run.

This feature is especially helpful to new coders. By utilizing it they are able to understand the ProIV timing cycle quicker than with other ProIV environments. Seasoned developers also find this view useful as a way of getting the grand overview of the function.

Undo/Redo

Traditionally, when editing ProIV code, once the user types a line and moves away from that line, the change was immediately committed to the code base. With IDE changes are only saved back to the code base (on the server) when the user specifically presses 'save'. There is also a window that tracks all changes made and gives the user the ability to undo and redo any changes recently made. This is typical of most windows applications, but something that has never before been available in such a way to the ProIV developer.



Column Design

In the function grids window, for example the 'Cycles' window, it is possible for the developer to add/remove and re-position all the columns to fit their needs. Cycles have numerous properties / columns, but a lot of them aren't needed most of the time. These can be hidden so that they don't clutter up the display. All properties are always displayed in the 'Properties' window though, either sorted alphabetically or by groups.

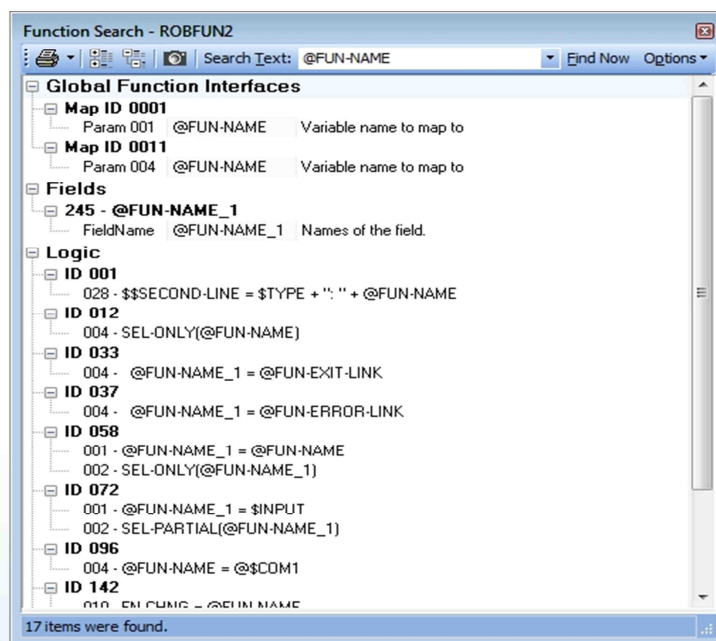
This column design allows the developer to design/shape ProIV IDE to how they want to use it, meaning each developer gets an environment they are comfortable with and ProIV IDE can look radically different to each developer.

These settings can all be saved and later loaded in profiles with a click of a button.

Profiles can help greatly in a training environment, as the trainer can setup different profiles that show different parts of the function or columns as and when that part is being taught.

Searching (In Function)

The search window, searches through all elements of the function (logic, screen fields, global interfaces etc.) and displays the results in a window.



Clicking on an individual result, makes ProIV IDE move to that place in the main window, so the code around the search result can be viewed.

This is an extremely useful tool in finding out how certain variables, fields or files are used within a function.

Once a search is done the user can press the snapshot button to keep the results in a window, while they then search for further items. This all makes searching very simple and allows the developer to still be able to edit the function while viewing the results.

Lists

Lists are an easy way to group certain functions, so that later they can be manipulated. The user can create lists manually or automatically from some of the other utilities in ProIV IDE.

Once a list is created the following can be done.

- Regen
- Use as a filter for File/Function Xref
- Use as a filter for Search
- Print

So they can build a list based on a File Xref, and then search all those functions with the search option for a certain field in logic.

This can considerably reduce the amount of time it takes to find what they are looking for, and can be a powerful tool for finding exactly what needs to be changed in a large system.

Search (System wide for logic)

Most of the ProIV code is done in logic. So this is a utility that will search through all the logic in the system (or restricted by a list, or partial search). Filters exist to allow removal of unused functions, utilities or other functions that are not part of the main system, to allow the user to centralize their search on the relevant part of the system.

Once the results are displayed, each item can be double clicked, and the relevant function will be loaded up into ProIV IDE.

Options exists also narrow the search to Global Logics, Comments or Case Sensitive searches.

File/Function Xref

This utility will list all the functions that use a specific file, displaying all the results in a formatted window that can then be used to load up the functions. The same filters and lists that can be used in the Search utility are also linked into this utility.

Function	Description	Type	Cycle	Seq	File Name	Mode A	Mode C	Mode D	Mode L	OneTime
ROBSCR	ROB'S SCREEN PLANNER	Screen	06	01	ZFUNCTDE	L	L	L	L	Y
ROBSCR	ROB'S SCREEN PLANNER	Screen	07	01	ZFUNCTDE	L	L	L	L	Y
ROBSPL	ROB'S SCREEN TO CHANGE \$SPOOL OPTIONS	Screen	01	02	ZFUNCTDE	L	C	L	L	
ROBULOG	ROB'S REMOVE UN-USED LOGICS	Update	03	01	ZFUNCTDE	L				Y
ROBUMSGA	ROB'S ADD UMSG TO ALL LOGICS	Update	01	02	ZFUNCTDE	C				
ROBUMSGD	ROB'S DELETE UMSG FROM LOGICS	Update	01	02	ZFUNCTDE	C				
ROBUMSGS	ROB'S LOGIC UMSG SCREEN	Screen	01	01	ZFUNCTDE	L	L	L	L	Y
ROBWINC	ROB'S WINDOW COPIER	(G) Screen	01	01	ZFUNCTDE	L	L	L	L	Y
ROBWINC	ROB'S WINDOW COPIER	(G) Screen	01	03	ZFUNCTDE	L	L	L	L	Y
ROBWINC	ROB'S WINDOW COPIER	(G) Screen	02	01	ZFUNCTDE	L	L	L	L	Y
ROBWINU	ROB'S WINDOW COPIER	(G) Update	02	01	ZFUNCTDE	L				Y
ROBwLOG	ROB'S WHERE IS LOGIC FINDER	(G) Screen	01	01	ZFUNCTDE	L	L	L	L	Y

Found 375 References.

Options exist also to narrow the Xref to certain file modes, clear flags and also to include alternate file definitions in the Xref.

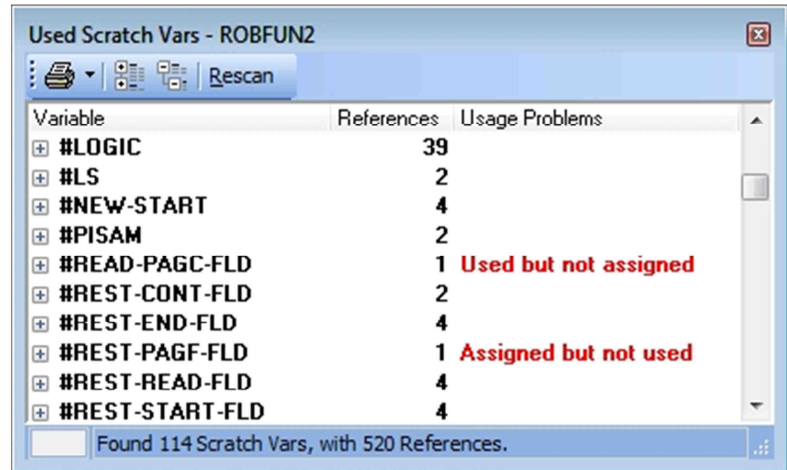
Display Used...

These utilities show where objects are used and the frequency of their use in a function.

They help a developer find certain objects in the code quickly and easily.

Scratch Vars

This window displays all scratch variables in a function, and where they are used. Also next to each variable can be a note if the variable has a problem, like being used but not assigned. This is extremely useful for finding typing / spelling mistakes in variable names, and helps prevent many bugs from leaving the development area.



Variable	References	Usage Problems
#LOGIC	39	
#LS	2	
#NEW-START	4	
#PISAM	2	
#READ-PAGC-FLD	1	Used but not assigned
#REST-CONT-FLD	2	
#REST-END-FLD	4	
#REST-PAGC-FLD	1	Assigned but not used
#REST-READ-FLD	4	
#REST-START-FLD	4	

Found 114 Scratch Vars, with 520 References.

Other objects displayed are 'Value Variables', 'Global Logics' and 'Global Functions'.

Printing

Printing is sometimes required to be able to visualize a large function.

Any Function, File, Global Logic or Task can be printed in full (including Logic colouring). Also any lists, searches or Xfers can be printed.

Most windows have a Print icon that allows the contents to be printed.

Dead Code

This utility lists all the code in a function that isn't actually called. This window can be used to identify the code and allow the developer to easily remove it, making the function more readable and easier to edit in the future.

QA Checks

This utility performs a list of over 60 checks on the code, to make sure the developer hasn't made technical mistakes in the code that can cause bugs, difficult code to follow or code that isn't actually doing anything.

It helps to maintain the ProIV code in a good state, and saves on time as any potential problems that may have been picked up later at the QA stage are found immediately and can be rectified by the developer.

Function Literals

This utility lists all literals in the function (text in quotes or numbers).

It helps as a check to make sure literals are correct. For example, maybe a certain function uses the code 'CORR' a lot in the function and there is a lot of code checking this text. This window would 'highlight' if there are any places in the function where the text had been typed incorrectly, say as 'COR'. This will help reduce bugs at development time.

Auto Updates

ProIV IDE on startup checks to see if there is a new version available, and if so, automatically downloads and installs it.

This can reduce support and maintenance costs, and also makes sure all your developers have the most current version.

This option can be disabled server side, if needed.

Bootstrap file locks

When a developer gets a record lock on one of the source files (i.e. Bootstrap files), the only information they get is that it's locked, and they just have to wait until the other user removes the lock. Other developers can lock the source code sometimes for hours if they have many sessions open, and leave one idle.

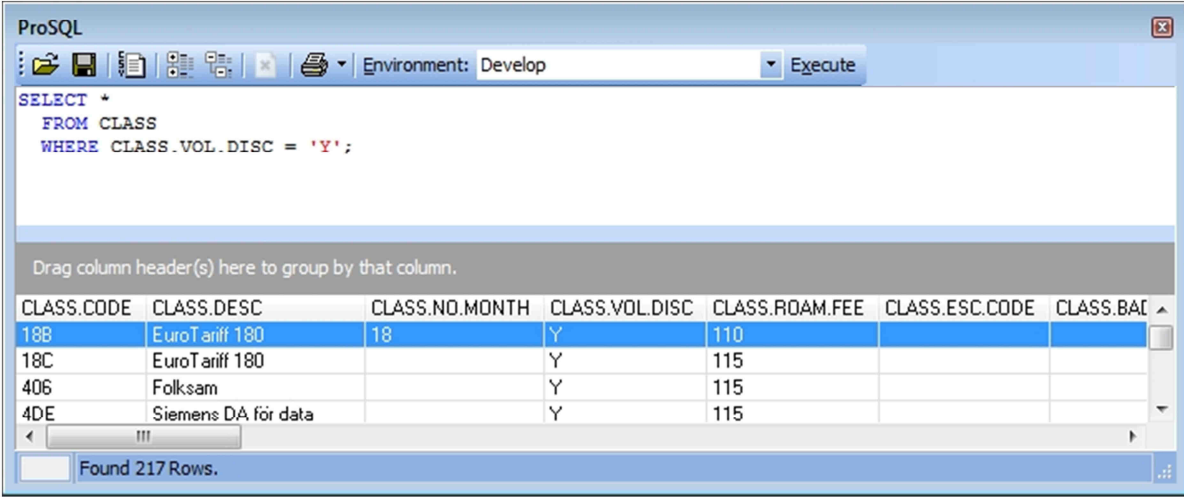
This window displays all the record locks on the system, and which developers has each one, time of the lock and what Process ID and terminal. This allows the developer to contact the other developer and request they unlock the file.

ProSQL

This is a very powerful function creator.

It converts standard SQL SELECT statements into ProIV functions that can extract data out of any of your applications data files.

It's specifically targeted at Production Support type roles, allowing data to be found using common SQL syntax.



The screenshot shows the ProSQL application window. The title bar reads "ProSQL". The interface includes a toolbar with icons for file operations and a dropdown menu showing "Environment: Develop" and an "Execute" button. The main text area contains the following SQL query:

```
SELECT *
FROM CLASS
WHERE CLASS.VOL.DISC = 'Y';
```

Below the query is a table with the following columns: CLASS.CODE, CLASS.DESC, CLASS.NO.MONTH, CLASS.VOL.DISC, CLASS.RQAM.FEE, CLASS.ESC.CODE, and CLASS.BAI. The table contains four rows of data:

CLASS.CODE	CLASS.DESC	CLASS.NO.MONTH	CLASS.VOL.DISC	CLASS.RQAM.FEE	CLASS.ESC.CODE	CLASS.BAI
18B	EuroTariff 180	18	Y	110		
18C	EuroTariff 180		Y	115		
406	Folksam		Y	115		
4DE	Siemens DA för data		Y	115		

At the bottom of the window, a status bar indicates "Found 217 Rows."

Once the data has been found, it can then be saved in CSV format and used in other programs like MS Excel.

It can also help developers and testers, as it makes it easy to find data that they need to check or create.

It saves large amounts of time, in that the developer/tester/production support doesn't have to write a ProIV function to find the data, it's created automatically from the ProSQL for them.

Conclusion

ProIV IDE has been written to make development easier and quicker.

All navigation has been optimized. The utilities and options have been put in places that are not only logical/valuable, but they are also easy to access.

It's easier to view and understand a function (and the whole system) in ProIV IDE than it is any other ProIV editor currently available.

Using ProIV IDE will significantly decrease your development times, improves quality and time to market. It helps the Business Analysts to analyze and specify changes to your system with the minimum of fuss.

Production support and data analysis can be helped also through the use of ProIV IDE and ProSQL.

Technical specs

Visual Basic 6: 72,000 Lines of code, 1000 controls, with 2500+ procedures.

ProIV Code: 10,000 lines of code in over 100 functions.

Server side: Proiv versions 4, 5 or 6

Client side: Windows XP, Vista and Win 7