Getting Started

Visual Guard 8.0

www.novalys.net

Table of Contents

About this document

Introduction

Welcome to the Visual Guard tutorial.

This guide provides you with a general description of Visual Guard. It describes how to implement security in a PowerBuilder application step by step.

The tutorial takes less than one hour and does not require any specific expertise.

Feel free to contact us if you have any questions regarding this tutorial.

System requirements

To run the programs contained in this tutorial, the following configuration is required:

- Windows 95/98, or NT 4.0 or 2000 or XP · PC Pentium 90 MHz
- 64 MB recommended (32 MB minimum)
- 40 MB of free space on your hard drive

Installation of Visual Guard is required.

Target users

This guide is for developers and project managers who want to secure PowerBuilder applications.

This tutorial does not require any special programming skills. However, for a more thorough understanding, familiarity with PowerBuilder's basic concepts would be helpful.

Visual Guard Presentation

Managing security in a PB application

The strategic importance of Client/server applications for businesses is increasing all the time.

Data confidentiality as well as restrictions on data updating are currently vital requirements.

VISUAL GUARD has been designed to meet the following requirements:

- Keeping sensitive data confidential: It is often overwhelmingly complex to implement effective security, either with the facilities provided by a DBMS or using PowerBuilder's (show/hide a field with respect to a user profile, display information in a special condition only, etc.).

- Customizing the application for each connected user (using a different business rule depending on the user profile, restricting the updating of certain data, etc.).

- Ensuring compliance with business rules when updating data: it is important to ensure that the only way a user can update data is by using the application designed for this purpose. A user could connect to the database using Desktop application tools (like MICROSOFT ACCESS, etc.) and update data without complying with business rules defined in the application.

- Centralizing security information in a common repository to preserve confidentiality and to allow for its updating without modifying the application.

- Easier security management. No technical knowledge is required.

- Easy implementation of security in existing applications as well as in new ones.

- No rule limitations: even the most complex rules are handled, not just hiding/showing buttons or menu options, etc.

Visual Guard Philosophy

Visual Guard has been designed to automate as much functionality as possible in order to manage security in a PowerBuilder application. This is performed at several levels:

Managing actions specific to PowerBuilder

Visual Guard manages the modification of key PowerBuilder object properties:

- Enabling/disabling a control (button, graph, listbox, etc.).).
- Making a control (button, graph, listbox, etc.) visible/invisible.
- Enabling/disabling menu items.
- Making menu items as well as their toolbar icons visible/invisible.
- Hiding/showing a datawindow column.
- Enabling/disabling a datawindow column
- Changing the WHERE clause of a datawindow sql statement.
- Filtering of rows displayed in a datawindow.
- Modifying text controls (Column names, contextual help, etc.)
- Modifying the properties of a datawindow element (data input restrictions, changing a column's validation message or validation rules, etc.).

Managing actions specific to your class libraries

Visual Guard has been designed to adapt to your PowerBuilder object class libraries. It can meet your specific requirements concerning security management.

You can use the Visual Guard's source code as an add-on to your specific objects (See the "Adaptations" chapter for further information).

Note: This tutorial uses a developed application which integrates PFCs.

Data-driven Security

It is often the case that the behavior of a window or an object has to be modified according to the data it displays and the connected user's profile. To do this, a solution which is currently used in applications is to create as many windows as there are data/profile combinations and to open the right window according to the connected user's profile. The first problem with this method is that tests on profiles are hardcoded in the application and this means that they cannot be modified without having to deploy a whole new application version. The second problem is that the application becomes much slower, upgrading it is more difficult (modifying security means deploying a new version) and this causes unnecessary complication.

Visual Guard has, therefore, been developed to dynamically manage the behavior of objects in accordance with the connected user's profile. This management is performed according to the data displayed in the windows, which allows for a very high level of abstraction when configuring application rules.

No rule restrictions

It is vital not to restrict the capabilities of security management. It may be the case that a section of the application requires very complex security management.

To do this, Visual Guard can trigger events (with their parameters) for your objects.

This feature works as follows:

- A window (or any object) in your application requires special security management, which cannot be processed automatically.
- The developer codes a user event, which manages this situation (this event can have parameters).
- Visual Guard will trigger this event when that window is opened and will pass the necessary arguments.

Principles of use

Visual Guard is used at two stages of an application’s lifecycle:

- When coding an application, the developer will define permissions. It will, therefore, create a permission repository.

- After deploying an application, a department manager will act as administrator. It will group permissions in order to create profiles. It will then assign profiles to users or groups of users.

Defining security

The developer will firstly define the application structure and a list of permissions to be attributed.

To do this, the application is broken down into 4 levels:

- The application itself.
- The entities managed by the application. An entity corresponds to a set of functions classified by theme. Example: "Client Management" or "Order Management."
- Functions related to an entity. Each function corresponds to an allowed action in your application. Example: "No viewing of an employee's personal details" or "Browsing of French clients only."
- The technical actions used to define a function. Example: "disable a menu", "hide a datawindow column."

Application management

Secondly the application is passed over to an administrator who will manage security.

To do this, it will define the following elements:

- **User:** physical person using the application. Example "John Smith"
- **Group**: group of users. A group is used to manage security for several users at the same time. Example: "Accounting Department user."
- **Profile**: The profile is a set of functions predefined by the developer which correspond to the user's role in the company. Example: the profile "Head Accountant" or "Secretary."
- The administrator can then assign each user to its corresponding profile.

Notice that all the tasks performed by the administrator **do not require technical knowledge.**

One must simply understand how the application works in order to define the rights of each user.

The application can, thus, be developed without technical know-how and without having to modify the application code.

You will be able to see this for yourself in this tutorial.

Using Visual Guard

Integrate Visual Guard into an application

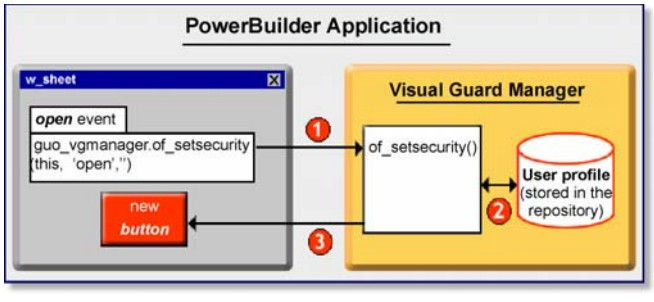
Visual Guard can be installed into new or older applications. Visual Guard stores information relating to security in a database. To do this, Visual Guard creates its repository in a database (Sybase, Oracle, Microsoft).

The developer must then integrate Visual Guard into his application

- The Visual Guard PBLs are added to the application. These PBLs are compiled and deployed with the application. Nothing else will be deployed on the customer's computer.

- To secure the application objects, the developer needs to add a call to Visual Guard (see diagram). This call is normally added to open event of the window ancestor. It is, of course, possible to call Visual Guard at any stage in the application.

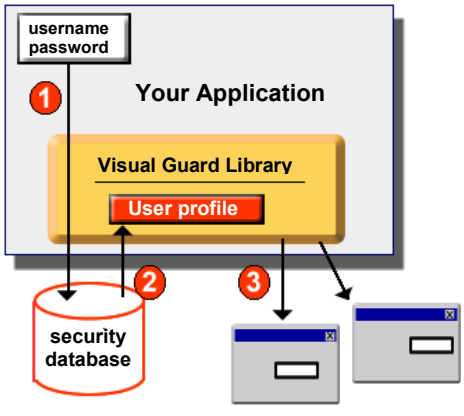
- Codes are not necessary to secure the application. You must define the security actions by using Developer Workshop. These actions are loaded with the user profile. Visual Guard then uses them to act on the application objects.



1. *When opening the window, w_sheet asks Visual Guard to securize it.*
2. *Visual Guard loads the actions to be performed for the User Profile.*
3. *Visual Guard executes these actions on the window (in this case, it disables the **new** button).*

How to load Visual Guard data

- When launching the application, the user gives his username/password.
- Visual guard connects to the security database and checks if the user is authorized to use the application and loads his profile. The profile describes the security actions of this user. It is stored in memory for the time the application runs. Then Visual Guard disconnects from the database (unless it is the database of your application).
- While the user navigates within the application, Visual Guard carries out the necessary security actions.



1. *The user runs the application.*
2. *Visual Guard loads the user profile from the security database and stores it in memory.*
3. *The user navigates in the application and Visual Guard applies security dynamically.*

Visual Guard Tutorial

About this tutorial

The tutorial has been designed to show how Visual Guard can be used to secure a PowerBuilder application. The tutorial is based on a demonstration application which already uses Visual Guard.

During this tutorial, we will first look at how new security function can be added to this application using the "Developer WorkShop". Then we will look at how to manage application users and to assign them user rights using the "Profile Manager."

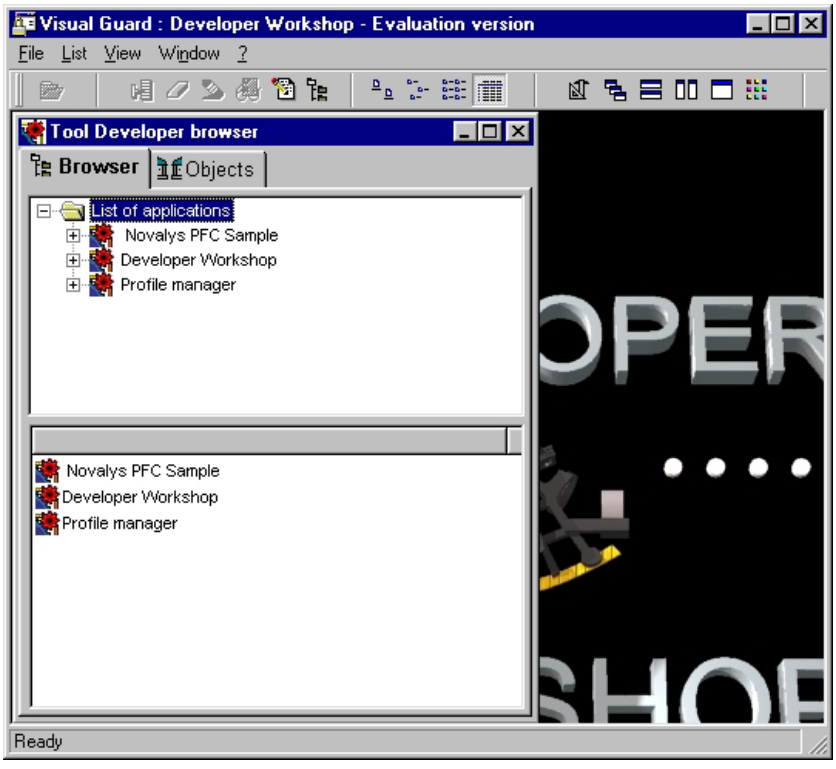
Developer WorkShop Presentation

The "Developer WorkShop" has been designed for developers. It defines actions to be performed on an application to implement security. In this chapter we will:

- Run the Developer WorkShop and get to know its features, such as entities, functions or technical actions.
- Go over the structure of an existing function used to secure the "***Novalys PFC Sample***." application in detail.
- Create two new functions.

Starting the Developer WorkShop

- Run the "Developer WorkShop" using the "Start" menu.
- Enter the userid "ADMIN" and the password "sql."
- The main application window opens and is displayed as follows:



The left window in the "Browser" tab shows the list of applications managed by Visual Guard. You will notice that Visual Guard can be plugged into several applications, which means that administration of all your applications can be centralized in a single repository.

For tutorial purposes, we are going to work with the "***Novalys PFC Sample***." This is an application developed using PFC.

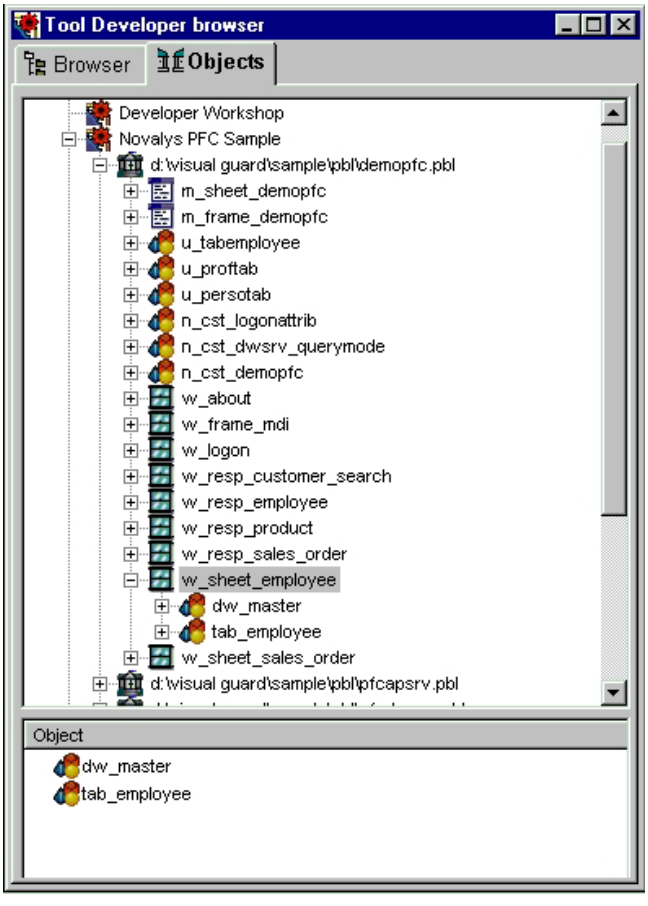
Visual Guard can, of course, be interfaced with any other framework or PowerBuilder application.

Viewing the objects in your application

The "***Objects***" tab is used to view the application structure. It enables you to graphically browse the application by analyzing the PowerBuilder code. This tab means less effort for the developer when creating actions to secure the application. To find out more about the interface, do the following:

- Click the "***Objects***" tab
- Deploy the contents of the "***Novalys PFC Sample***" item. You can browse within this application and view the various application objects.

- Select the management window for employees "***w_sheet_employee***" located in the pbl "***demopfc.pbl.***" This is the window in which we are going to work.



Opening an entity

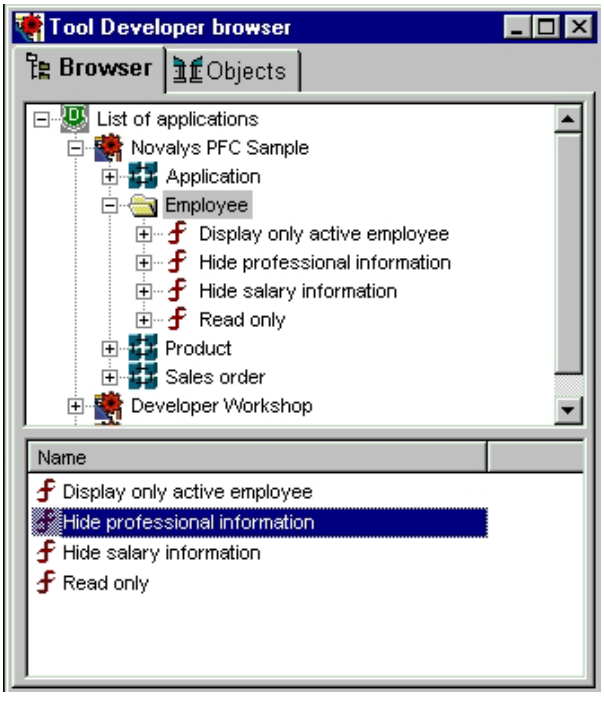
Now we are going to browse the application entities. Visual Guard maps entities to a set of functions classified by theme. Entities have two important roles:

- They are used to manage applications containing a large number of security functions by partitioning them into several themes, which can be easily maintained by users.

- They provide the user who manages application security with a business overview of the different functions.

You can browse the entity "***Employee***" which we are going to use, by performing the following actions:

- Click the "***Browser***" tab
- Deploy the node of the "***Novalys PFC Sample***." The list of application entities appears.
- Deploy the node of the "***Employee***" entity. The list of functions declared for this entity is displayed.



Examining a function

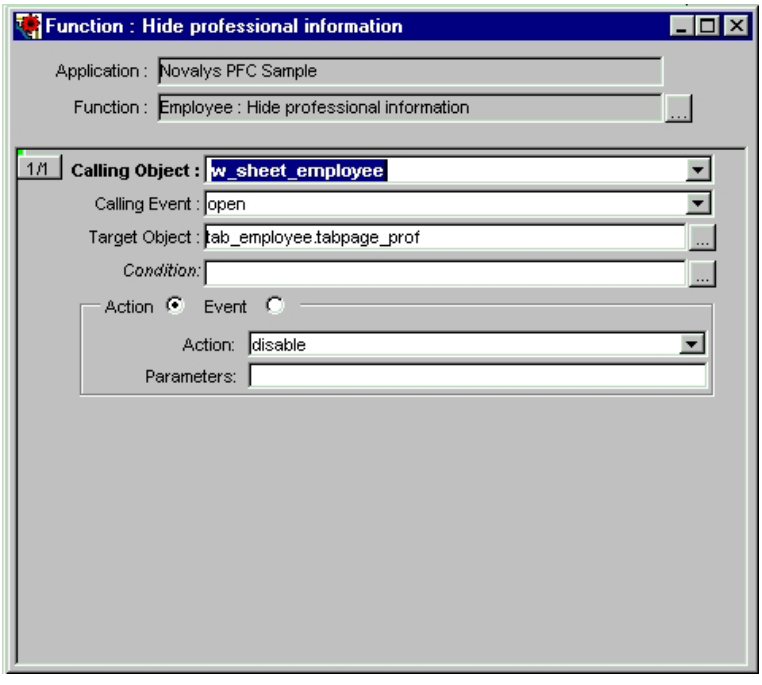
We are now going to browse a function. With Visual Guard, functions correspond to the privileges that the administrator can assign or cannot assign to an application user. Later we will see how to assign these privileges in the "Profile Manager."

For the moment we are going to see how the developer sets, using the Developer WorkShop", the technical actions to be performed to implement these rights in the application.

To do this, we are going to browse the function ***"Hide professional information"*** which already exists and try to examine it in detail.

- Position the cursor on the function "***Hide professional information***" of the "***employee***" entity. This function is designed to hide the professional information of an employee from certain users.

- Open the property sheet of this function by clicking the right mouse button on the item and selecting the menu item "***Open.***" A new window opens on the right containing information on that function.



Notice how the function is made up of a single action. Visual Guard will trigger this action on the "***tab_employee.tabpage_prof***" object when the window "***w_sheet_employee***" is opened. Visual Guard will disable this tab (which contains the professional details of an employee).

Several technical actions can of course be associated with a function. If you browse the ***"Read only"*** function of the entity "***employee***", you can see that access to employee management in read-only mode requires a list of technical actions in the application.

Creating your first function

As an example we are going to create a function similar to the one defined above, which will be used to hide the column containing salary information. Do the following:

- Click the "***browser***" tab in the left window. Select the entity "***employee***". The list of functions for this entity is displayed in the list below the entity tree.

- To add a new function, right-click this item and select the menu item "***Add.***" A "***New function***" item will be added to the end of this list. Replace it with: "***Hide salary***"

- Right-click this item and select the menu "***Open***". The corresponding ***function*** opens.

- To add an action, select the sub-menu "***Add action***" from the "***List***" menu in the main menu bar. A line of action appears in the list of actions.

- To enter parameters for this action, select the tab "***Objects***" in the left window. Position the cursor on the item "***w_sheet_employee***" in the pbl "***demopfc***". Drag it using the mouse to the "***Calling object***" column. You have just associated this action with the "***w_sheet_employee***" window.

- In the calling event column, select the "***open***" value from the list. You have just indicated that this action will be performed during the opening of the "***w_sheet_employee***" window.

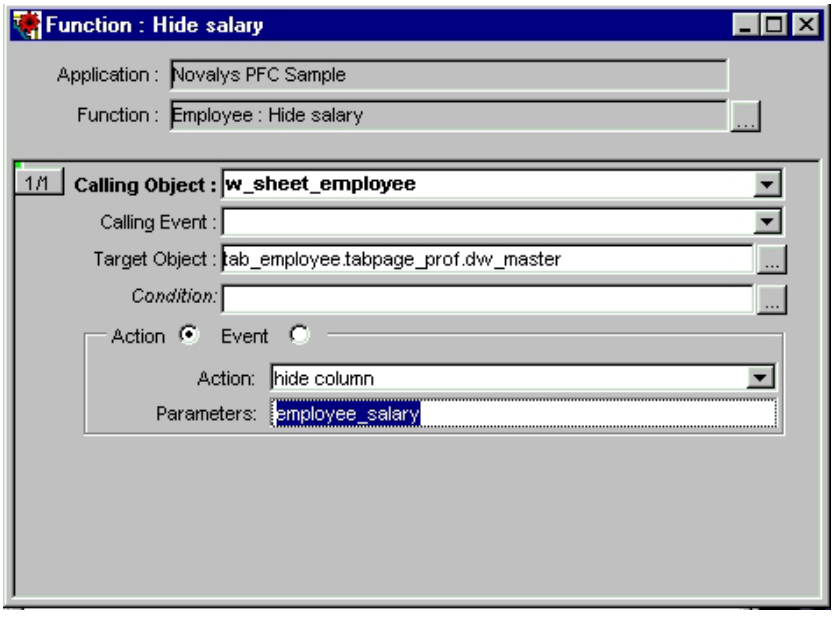
- In the "***Objects***" tab in the left window, deploy "***w_sheet_employee***", "***tab_employee***", ***"dw_master"***. Drag the item "***dw_master***" and drop it in the "***target object***" column. This indicates to Visual Guard that the action will be performed on this object.

- In the "***action***" area, select the value ***"hide column"*** from the list to indicate that Visual Guard will make the object invisible.

- In the "***Objects***" tab, deploy the node ***"d_ff_employee_professional"***. Drag the item ***"employee_salary"*** and drop it in the ***"parameters"*** column to indicate the column which will be invisible

- Save the modifications by selecting the item "***Save***" from the "***File***" menu on the main menu bar.

You should obtain the following result:

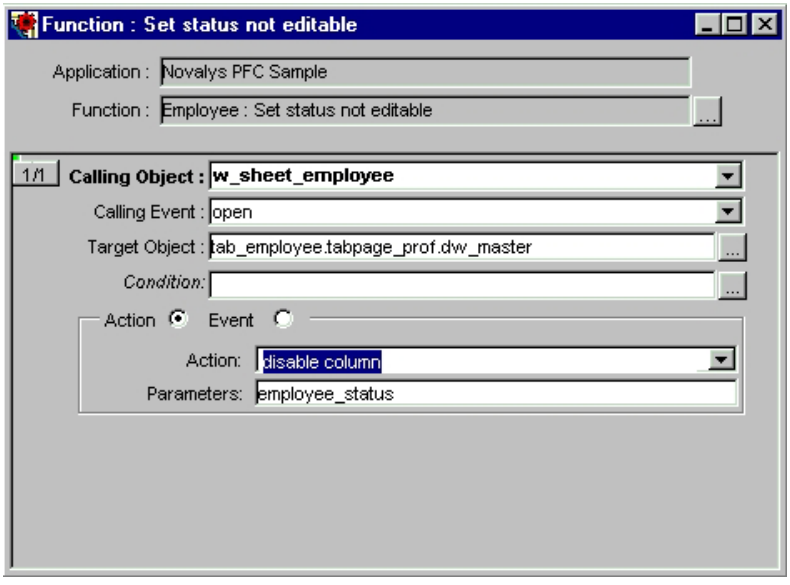


Creating a second function

The **status** of an employee is an important professional detail (Active, terminated, On leave). In this example, inserting information into this field is restricted for *certain* users. We are going to create the function, which will be used to disable the corresponding column. Do the following:

- Create a function "***Set status not editable***" for the entity "***Employee***". To do this, follow the previous example.
- Open the function you have created.
- Add an action for the window "***w_sheet_employee***" This action should disable the ***"employee_status"*** column in the ***"tab_employee.tabpage_prof.dw_master"*** datawindow.

You should obtain the following result:



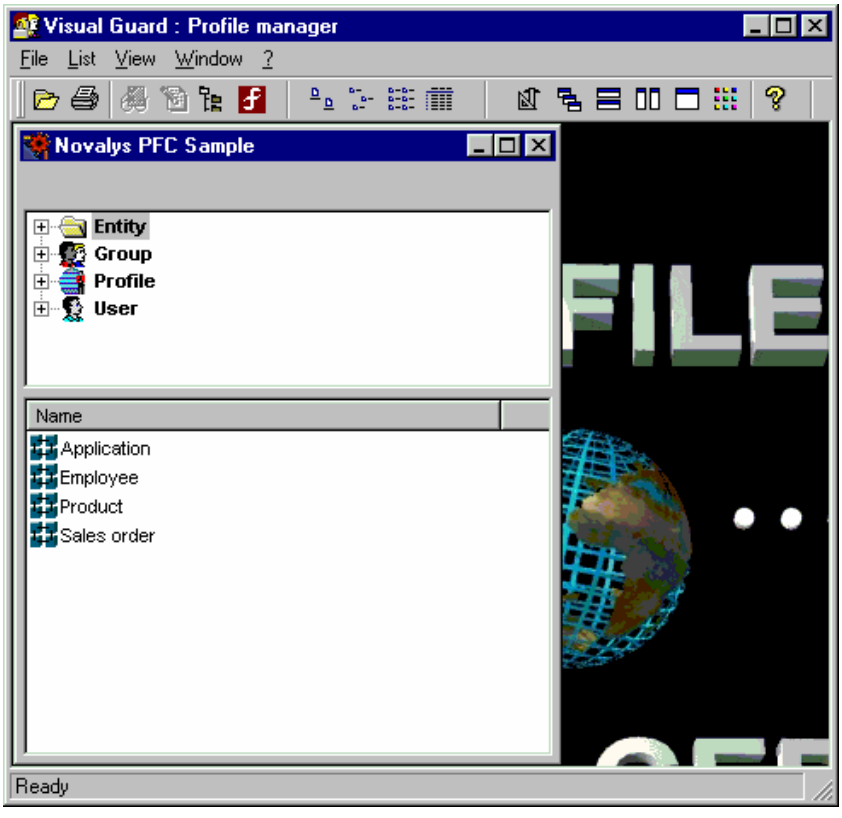
Profile Manager Presentation

The "Profile Manager" enables a security manager to create users and to assign them privileges. To see its features, do the following:

- Start the "Profile Manager" and get to know the different features that it implements, such as profiles and groups of users.
- Create a new profile using the two functions you have just created.
- Assign this profile to a user.
- Test this new profile in the "***Novalys PFC Sample***".

Starting the Profile Manager

- Enter the userid "**admin**" and the password "**sql**."
- A window with the list of managed applications appears. Select the "***Novalys PFC Sample***" and click on "***OK***." The main window is displayed as follows:



The left window displays four main objects that the application administrator will manage:

- Entity: They are the same as defined in the "Developer WorkShop". If you deploy the nodes located under the "entity" node, you will find the same list as that in the Developer WorkShop as well as all the functions attached to these entities. Notice how technical actions have completely disappeared. The purely technical aspect of security is transparent to the administrator.

- Group: When an application is accessed by a large number of users, user privileges administration can become very complex. To handle this, you can group these users and manage them all at the same time.

- Profile: With Visual Guard, profiles are used to define a set of privileges to an application. When the administrator wants to assign the privilege to use an application to a user, he assigns a profile to this user.

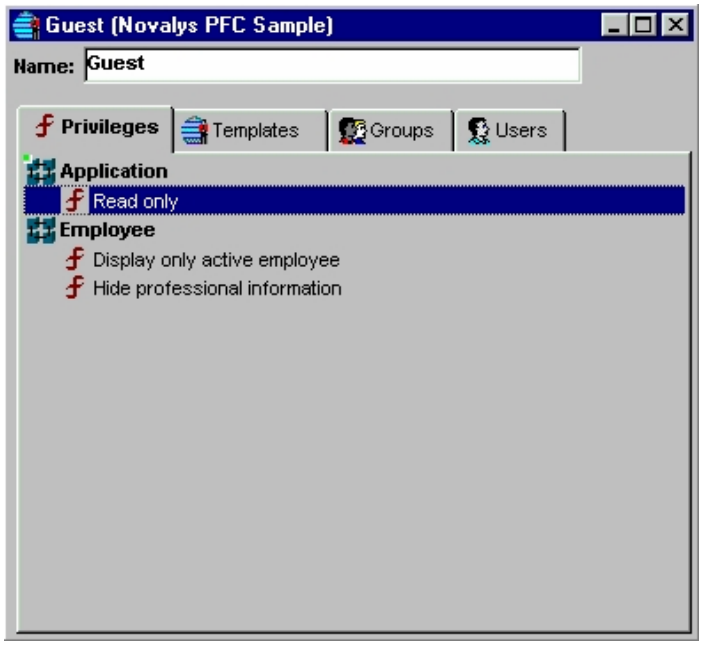
- User: In Visual Guard, users are not simply assigned to an application. They are assigned to all the applications managed by Visual Guard. This simplifies user management.

Examining an existing profile

We are now going to examine profiles more closely. Then we are going to assign a user to a profile. To do this, we are going to browse the profile ***"Guest"***. Users who have been assigned this profile have read-only access to the information and may only consult personal information pertaining to active employees. To browse this profile, do the following:

- Deploy the node "Profiles" then the node ***"Guest"***.

- Right-click this item and select the menu item ***"Open"*** The corresponding profile opens:



- In the "***Privileges***" tab, the functions associated with the profile are displayed. In our example the profile is composed of 3 privileges, "***Read only***", ***"Display only active employee"*** and "***Hide professional information***".

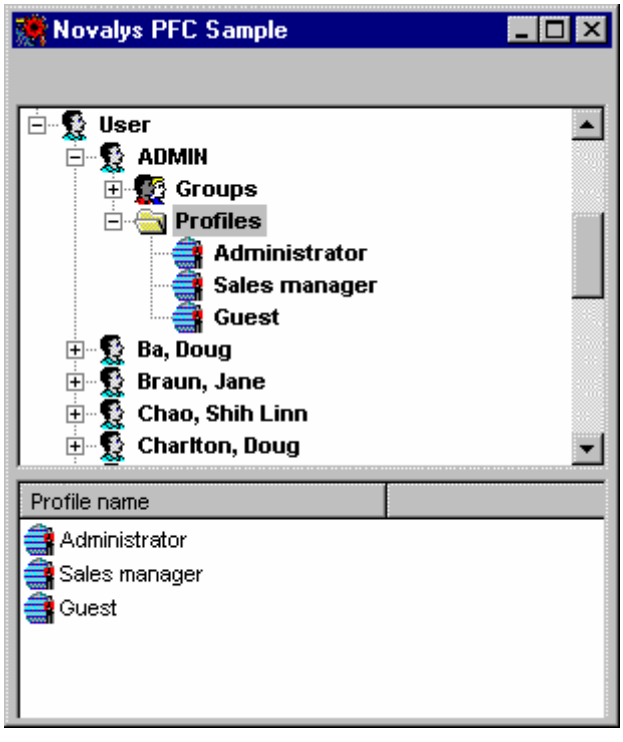
- The "***Templates***" tab indicates that this profile reuses the privileges defined in the template profile(s) (a computer expert would say that it inherits from one or several profiles). The templates make the management of different profiles easier when security problems are complex. Templates will not be used in our example.

- The "***Groups***" tab indicates the groups of users assigned to this profile. There is none for this profile.

- The "***Users***" tab indicates the users assigned to this profile. In the left browser, deploy the node "***users***" and select the user "***ADMIN***" Drag the item "***ADMIN***" and drop it in the list of users in the user tab. You have just assigned this profile to the user "***ADMIN***."

- Save the file by clicking "***File"*** and then ***"Save"***.

- Check that the user "***ADMIN***" now has 2 profiles. To do this, deploy the node "***Profiles***" in the node "***ADMIN***" in the left browser. The user has 3 profiles: ***"Administrator", "Sales Manager"*** and "***Guest***". You should obtain the following result:



Testing existing profiles

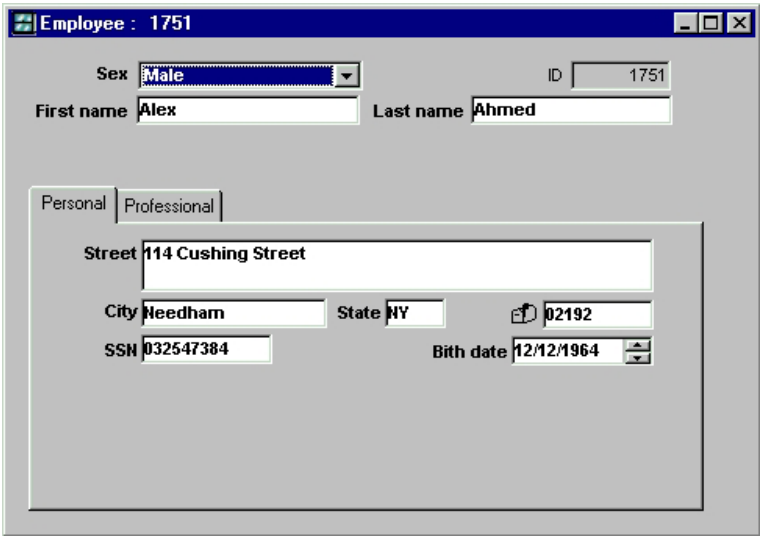
We are now going to examine how the application behaves according to the profiles assigned to a user.

- From the "***Start***" menu in Windows, run the application "***Sample***."

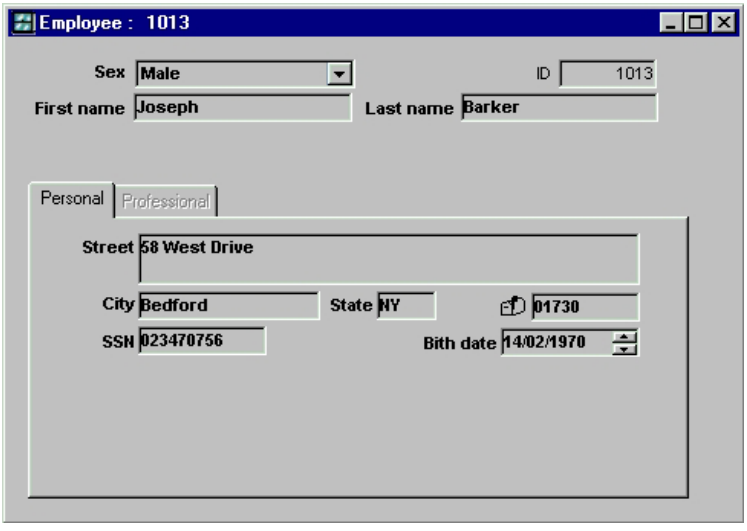
- In the connection window, enter the userid "***ADMIN***" and the password "***sql***" This user has 2 profiles; Visual Guard opens a window allowing you to select the profile to be used. Select first of all the profile "***Administrator***". The main window opens.

- Select the menu item "***File/Open/Employee***". A window will open. Click on the "Run!" button. The list of employees will appear. You will notice that all employees have been displayed regardless of their status. Select the first one on the list and click on the "OK" button.

- The employee sheet opens. The first tab contains the employee's personal information. The second tab displays professional information including the salary. You should obtain the following result:



- Close the application and restart it by entering the userid "***ADMIN***", password "***sql***" and then by selecting the profile "***Guests***". Open the employee sheet again as before. Note that, in the list of employees, only the active employees are visible. The employee page is read-only and professional information is not accessible.


Employee : 1013
Sex
Male
ID
1013
First name
Joseph
Last name
Barker
Personal
Professional
Street
58 West Drive
City
Bedford
State
NY
01730
SSN
023470756
Bith date
14/02/1970

Conclusion

Congratulations! You have completed the Visual Guard tutorial.

Visual Guard has been specially designed for security management within a company. It includes a repository for all applications. It simplifies user and profile management by using groups of users and profile templates.

Visual Guard makes application development easier, by separating security administration from development, since there is no need to redefine the application's security policy each time.

If you would like to learn more about Visual Guard, you can create new functions or profiles on the Novalys PFC Sample. If you would like to see how Visual Guard is integrated into an application, the Novalys PFC Sample application source codes may be consulted in the ***"Visual Guard/Sample/pbl"*** repertory. The following is the list of scripts which implement the code necessary for integrating Visual Guard:

Event or Function	Description
Object: n_cst_appmanager (pfeapsrv.pbl) Function: of_setsecurity	Enables or disables the security service
Object: n_cst_appmanager (pfeapsrv.pbl) Event: pfc_logon	Authenticates the user and loads security information.
Object: u_dw (pfemain.pbl) Event: constructor	Establishes security for a datawindow
Object: w_master (pfemain.pbl) Event: pfc_open	Establishes security for a window
Object: w_master (pfemain.pbl) Event: pfc_postopen	Establishes security for a window after loading data
Object: vge_n_cst_vgmanager	Extends standard actions and adds a new specific action

This documentation is about the tutorial of the following software :

Visual Guard

IDDN.FR.001.050019.01.S.P.2000.000.10600

This document is provided for information only.

Information in this document is subject to change without notice.

IDDN.FR.001.050019.01.S.P.2000.000.10600

NOVALYS S.A

41/43 rue Paul Bert

92 100 Boulogne Billancourt

France

Tel.: (+33) 1.41.31.82.82

Fax : (+33) 1.41.31.82.90

Visit our web site : http://www.novalys.net