FL MMOG
5.7 Protection version
User Manual

1.4
17.11.2009

www.star-force.com

Contents

1 The FL MMOG product

Продукт The FrontLine MMOG product is the complex aimed at the protection of MMO games against a range of threats: FL MMOG provides protection of the game servers against reverse engineering, modification and running servers on unauthorized sites; protection of the game code against reverse engineering and cracking; protection against running and executing cheat programs and bots; protection of traffic between an MMOG server and a client.

FL MMOG protection features:

- Complicates non-invasive and invasive attacks;
- Has no significant impact on the game operation;
- Simple protection implementation;
- No binding to any object (optical media or computer) is required;
- No activation is required.

The FL MMOG solution allows:

1. **Checking the integrity of the application during its start**.
   Such check is designed for additional protection of the protected game modules against modification. To do so, digital signature of the protection is added to a module. The protection system checks this signature in its loader during module loading (see **Protection API functions**). The check is performed automatically when the game runs. Besides, the protection operates if executable modules of the game are infected.
2. **Checking the integrity of code and data in memory**.
   Controlling the integrity of read-only code and data in memory. This method is designed to check a protected executable file during game operation. The integrity of the read-only elements of the protected file is checked, i.e. game variables are not checked. To perform the check, the protected game calls the appropriate SF API function (see **Protection API functions**). This operation greatly complicates the modification of the protected file during game operation.
3. **Protection of certain game variables against unauthorized access/modification**.
   To protect variables, 'secure classes' from the SF API (SF Uint) are used instead of the built-in classes (Uint) (see **Applying secured classes**). This protection greatly complicates the modification of the game variables during game operation.
4. **Checking the integrity of the data files**.
   Checking the integrity of read-only files (the protected game checks the digital signatures of the data files). The check is performed via StarForce API (see **Protection API functions**). This operation complicates analysis and modification of the data files of the protected game.
5. **Checking the parent process**.
   This method (enabled by default) allows running the protected game from the authorized processes only. If a process (a file of malicious software) is in the black list, an error is displayed. Such files are checked according to signatures. The black list can be supplemented with time.

6. **Protection of traffic between client and server**.
   This method consists in encrypting traffic between game client and server and greatly complicates traffic modification or substitution. The main point of the protection is that the communication channel between client and server is cryptographically protected. Installation of an additional StarForce Java-module on the game server is required to implement the protection (see **Protection API functions**).

To perform an application protection, the Protection Studio software system is used. Protection Studio is one of the components of StarForce (SF) protection system.

This manual describes the steps to perform protection, and the protected application testing and maintenance.

Notational conventions

This manual uses the following conventions:

Convention	Description
`Parameter`	Code
Service	Command, button, menu, tab
General options	Protection Studio section, product name
`Binding`	Field, parameter in Protection Studio; parameter value
`File`	File, folder name
Main parameters	Group of fields, parameters
Information	Definition; important note
Attention!	Important information

Notational conventions

2 The scheme of implementing FL MMOG protection during the product development and release

#	Stage	Manual section(s)
1	**Creating a protection project file***	**'Project' section**
	Game development stage	
2	**SDK generation**	**StarForce SDK**
3	**Embedding the required functions into the protection code**	**Protection API functions**

#	Stage	Manual section(s)
4	**Testing the unprotected client application**	
	Protection stage	
5	**Selecting and adding files for protection to the project file**	**Files for protection**; **'Files' section**
6	Selecting functions to protect and specifying the methods of their protection	'Methods' section
7	Setting the parameters	'General options' section; 'Advanced options' section
8	Setting protection graphical skin	'Protection GUI options' section
9	Specifying the path to a folder to download files from the protection server in	Output folder
10	**Protection of files**	**Protection of files**
11	**Preparing the product installation package**	**Creating the product installation package**
12	**Testing the functionality of the protected software product****	**Testing the protected application**

Protection implementation procedure

***** Mandatory actions are in boldface.

** When getting negative testing results, repeat the stages 2, 3, 4, and further.

3 Product protection and testing

3.1 Getting started with protection

3.1.1 Installing Protection Studio

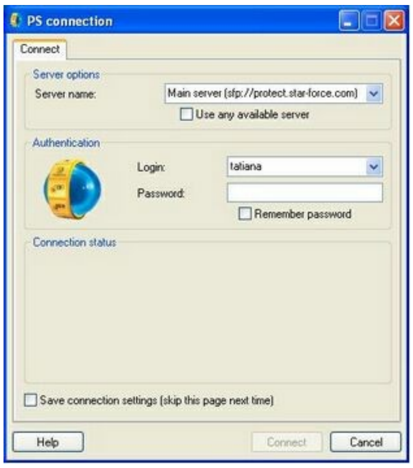
The Protection Studio installation file can be downloaded from StarForce website: http://www.star-force.com. To do so, enter the **`Members Area`** on the website using the login and password of the account provided.

An **account** is created in Protection Studio informational space for every user who starts working with the StarForce protection system. The unique login and password of the account are provided to the user by StarForce customer support.
Account is used for:

- entering the Members area of StarForce website, intended for clients in order for them to download all necessary software and documents for protection.
- running Protection Studio with connection to workspaces the account is assigned to.

The installation is performed in a usual manner.

3.1.2 Running Protection Studio


PS connection

Connect

Server options

Server name: Main server (sfp://protect.star-force.com)

☐ Use any available server

Authentication

Login: tatiana

Password:

☐ Remember password

Connection status

☐ Save connection settings (skip this page next time)

Running Protection Studio

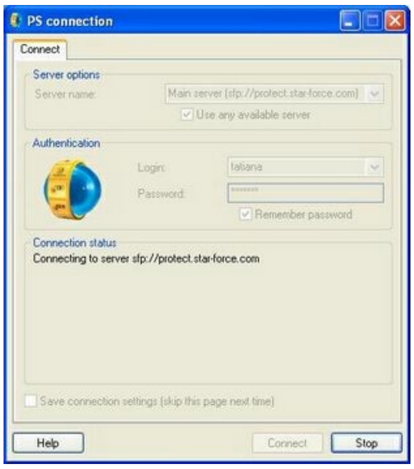
After the installation Protection Studio is run from the **Start** menu by the following command: **Start|Programs|Protection Studio 2|Protection Studio**.

The “PS connection” window is displayed then (see above).

On the **Connect** tab, perform the following operations:

1. Specify or select the required server in the **`Server name`** combo box, or check the **`Use any available server`** checkbox, if the connection to any available server is appropriate.
2. Enter login and password of the account. If the **`Remember password`** checkbox is checked, the password will be entered automatically in the future, when the corresponding login is typed or selected.
3. Press **Connect**.
   The connection process message is displayed in the **`Connection status`** log.
   If the **`Use any available server`** checkbox is checked, a single attempt to connect to each of the protection servers from the list is made by turn. If all the servers are busy, the user can try to reconnect by pressing **Connect**, or close Protection Studio by pressing **Cancel**, and try again later.

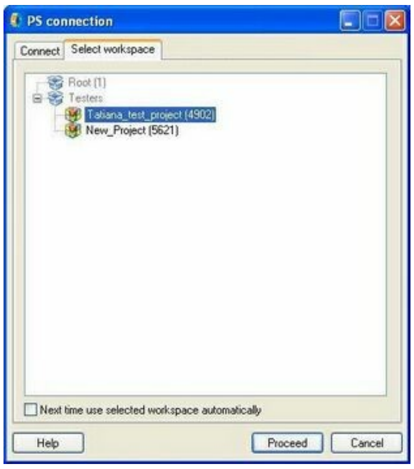
It is possible to stop the process by pressing **Stop** (see below).



Server options
Server name: Main server (sfp://protect.star-force.com)
✓ Use any available server
Authentication
Login: tatiana
Password: *******
✓ Remember password
Connection status
Connecting to server sfp://protect.star-force.com
Save connection settings (skip this page next time)

The connection is stopped

If it is not supposed to change connection details, specified in Protection Studio connection window for some time, the **`Save connecting settings (skip this page next time)`** checkbox can be checked. Then an attempt to connect to a server and to switch to the next window is made when running Protection Studio next time – as if **Connect** has been pressed.
After the connection to a server succeeds, the project selection window is displayed (see below).



Root (1)
Testers
Tatiana_test_project (4902)
New_Project (5621)
Next time use selected workspace automatically

Workspace selection

Select a workspace where the protection is to be performed, on the **Select workspace** tab.

Workspace is an aggregate of informational objects available for a user of StarForce protection system. A project is created by StarForce customer support after a StarForce sales manager is provided with the protection parameters selected by the client. A protection project contains default protection settings, as well as parameters related to workspace management.

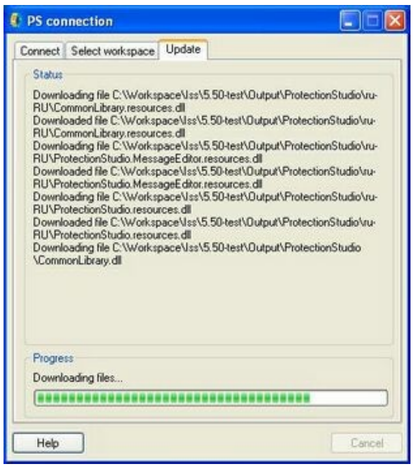
After a workspace is selected, press **Continue**.

If it is not supposed to change the selected workspace for a long time, the **`Next time use selected workspace automatically`** checkbox can be checked. Then a transfer to the next window is made automatically when running Protection Studio next time – as if

Continue has been pressed.

If only a single workspace is available to the user, it is selected automatically, that is, the project selection window is not displayed regardless of whether the **`Next time use selected workspace automatically`** checkbox is checked.

As a result, the automatic update window is displayed (see below).



```
Downloading file C:\Workspace\Iss\5.50-test\Output\ProtectionStudio\ru-
RU\CommonLibrary.resources.dll
Downloaded file C:\Workspace\Iss\5.50-test\Output\ProtectionStudio\ru-
RU\CommonLibrary.resources.dll
Downloading file C:\Workspace\Iss\5.50-test\Output\ProtectionStudio\ru-
RU\ProtectionStudio.MessageEditor.resources.dll
Downloaded file C:\Workspace\Iss\5.50-test\Output\ProtectionStudio\ru-
RU\ProtectionStudio.MessageEditor.resources.dll
Downloading file C:\Workspace\Iss\5.50-test\Output\ProtectionStudio\ru-
RU\ProtectionStudio.resources.dll
Downloaded file C:\Workspace\Iss\5.50-test\Output\ProtectionStudio\ru-
RU\ProtectionStudio.resources.dll
Downloading file C:\Workspace\Iss\5.50-test\Output\ProtectionStudio
\CommonLibrary.dll
```

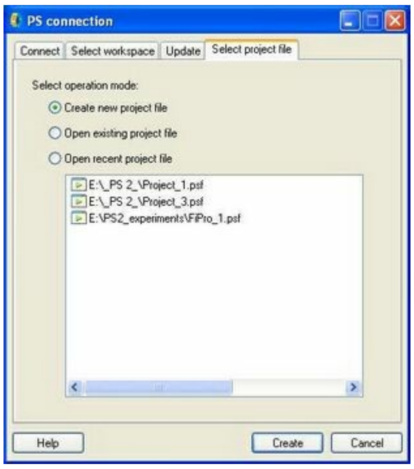
Downloading files...

PS update

Every time Protection Studio is run, the update check is performed (if the corresponding option is not disabled (see figure)). If there have been any updates, Protection Studio performs the corresponding changes automatically. Thus, you always work with the latest program version. Protection Studio restarts automatically after the update. This process takes place until the user disables this service (see **Protection Studio interface**).

Note. If the automatic update service is disabled, the **Update** tab is not displayed in the connection window.

If no update is required, the process switches to the project file selection tab automatically (see figure).



Select operation mode: [✓ Create new project file] Open existing project file Open recent project file

E:_PS 2_\Project_1.psf
E:_PS 2_\Project_3.psf
E:\PS2_experiments\FiPro_1.psf

Project file selection

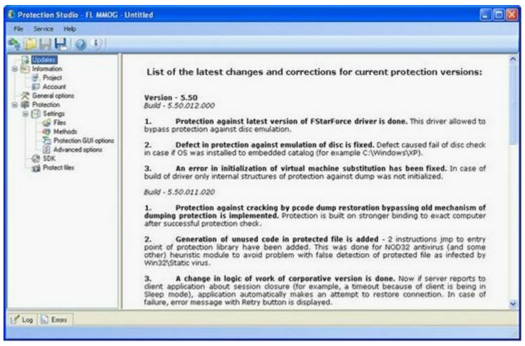
The **Select project file** tab allows selecting a project file to work with.
Project file is created in Protection Studio when working on product protection and is stored on the user's computer as a `.PSF`-file. Sometimes several `.PSF`-files can be created for

complicated projects.

There are three ways to open a file:

- *Create new project file*. After pressing the **`Create new project file`** option button, the **Create** button becomes enabled. After pressing it, the main program window with the default parameters (an empty project) is displayed (see figure).
- *Open an already existing project file*. After pressing the **`Open existing project file`** option button, the **Open** button becomes enabled. After pressing it, the standard dialog box of opening a file appears. Once the necessary file is opened, the main program window with the parameters of the opened project file is displayed (see figure).
- *Select a file from the list of files the user has already worked with*. The last five such files are listed in the center of the window. After pressing the **`Open recent project file`** option button and selecting the required file from the list, the **Open** button becomes enabled. After pressing it, a new program window with the parameters of the selected project file is displayed (see figure). Instead of performing the above-mentioned operations it is also sufficient to double click the name of a file in the list.

The main window of Protection Studio (see figure) is the interface, which is used to specify the product protection parameters available within the selected workspace. It allows running the protection services with the specified parameters as well.



List of the latest changes and corrections for current protection versions:

Version - 5.50
Build - 5.50.012.000

1. **Protection against latest version of FStarForce driver is done.** This driver allowed to bypass protection against disc emulation.

2. **Defect in protection against emulation of disc is fixed.** Defect caused fail of disc check in case if OS was installed to embedded catalog (for example C:\Windows\XP).

3. **An error in initialization of virtual machine substitution has been fixed.** In case of build of driver only internal structures of protection against dump was not initialized.

Build - 5.50.011.020

1. **Protection against cracking by pcode dump restoration bypassing old mechanism of dumping protection is implemented.** Protection is built on stronger binding to exact computer after successful protection check.

2. **Generation of unused code in protected file is added** - 2 instructions jmp to entry point of protection library have been added. This was done for NOD32 antivirus (and some other) heuristic module to avoid problem with false detection of protected file as infected by Win32\Static virus.

3. **A change in logic of work of corporative version is done.** Now if server reports to client application about session closure (for example, a timeout because of client is being in Sleep mode), application automatically makes an attempt to restore connection. In case of failure, error message with Retry button is displayed.

Protection Studio interface

3.2 Implementing protection

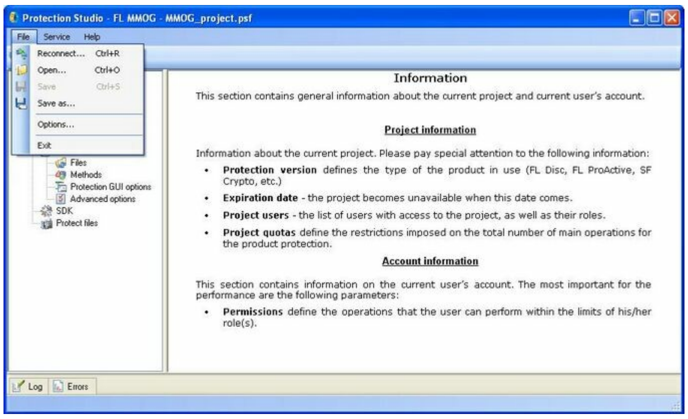
3.2.1 Protection Studio interface

The caption of Protection Studio window contains the name of the protection project, **FL MMOG** and the name of the opened project file. If a new project has been created and has not been saved yet, its name in the caption is **Untitled**. There is an asterisk '*' after the project name, until the results of its last editing are saved. The symbol vanishes after the **Save** or **Save as** commands are performed.

3.2.1.1 Menu commands

File menu

The **File** menu is shown in figure below.



Information

This section contains general information about the current project and current user's account.

Project information

Information about the current project. Please pay special attention to the following information:

- **Protection version** defines the type of the product in use (FL Disc, FL ProActive, SF Crypto, etc.)
- **Expiration date** - the project becomes unavailable when this date comes.
- **Project users** - the list of users with access to the project, as well as their roles.
- **Project quotas** define the restrictions imposed on the total number of main operations for the product protection.

Account information

This section contains information on the current user's account. The most important for the performance are the following parameters:

- **Permissions** define the operations that the user can perform within the limits of his/her role(s).

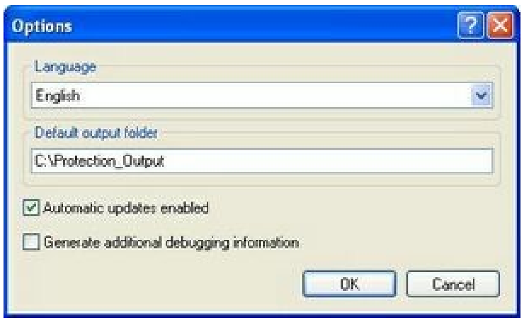
Protection Studio interface. File menu

Command	Purpose
Reconnect…	Returns to the connection window to select another protection server, workspace or project file. Protection Studio window is closed and the connection window is displayed again.
Open...	Opens a project file saved to the user's computer (.PSF).
Save	Saves current settings to a project file (.PSF).
Save as...	Saves current settings to another project file (.PSF).
Options...	Opens the 'Options' window.
Exit	Closes Protection Studio.

File menu commands

To provide safety, it is recommended to save changes in the protection parameters to a project file, although it is possible to perform protection without saving the changes.

Options are displayed in an additional window (see below).



Language: English

Default output folder: C:\Protection_Output

[✓] Automatic updates enabled

[ ] Generate additional debugging information

Opitons

In this window:

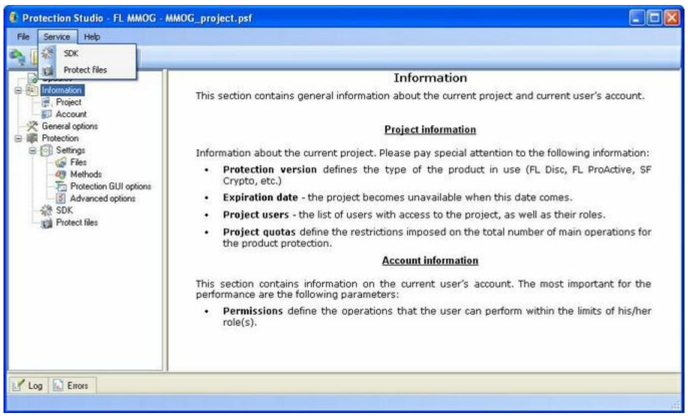
- It is possible to select Protection Studio interface language. Two languages are supported in Protection Studio: English and Russian. In order to change the language, it is necessary to select the **Options** command from the **File** menu (see figure), select the required language in the displayed window and press **OK**.

In order for the change to take effect, restart the application. The corresponding prompt message appears when pressing **OK** button after the language is selected.

- The default folder is specified in order to save the results for all the services. The folder can be redefined for each service in the corresponding section. If such redefinition takes place for a certain service (see Protection of files, SDK), the subsequent changes in the **`Output folder`** field in the 'Options' dialog box will not affect it.
- It is possible to disable the automatic update of Protection Studio, which is enabled by default (it is recommended to disable this option only after contacting customer support).
- It is possible to enable the generation of the additional debugging information, which is saved to the `debug.log` file in the root folder of Protection Studio. These data may be helpful if any protection implementation errors occur.

Service menu

Service menu contains links to the **SDK**, **Protect files**, **Generation templates** and **Generate Developer Utility** sections. The names of the links coincide with the names of the services displayed in Protection Studio (see below). Thus, a section that corresponds to a service can be accessed in two ways: from this menu or by selecting a tree element with the same name in Protection Studio window.



Information

This section contains general information about the current project and current user's account.

Project information

Information about the current project. Please pay special attention to the following information:

- **Protection version** defines the type of the product in use (FL Disc, FL ProActive, SF Crypto, etc.)
- **Expiration date** - the project becomes unavailable when this date comes.
- **Project users** - the list of users with access to the project, as well as their roles.
- **Project quotas** define the restrictions imposed on the total number of main operations for the product protection.

Account information

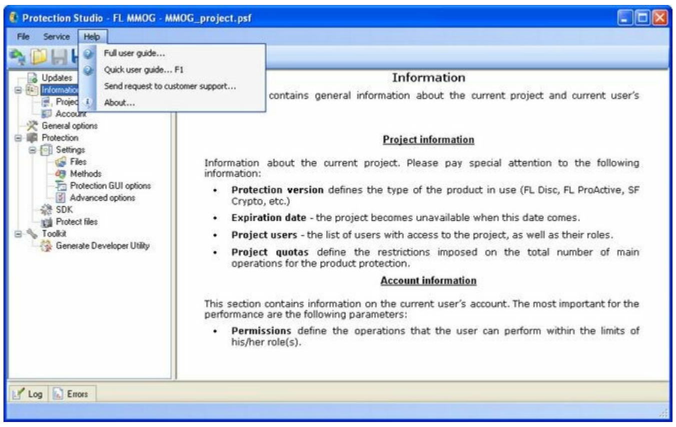
This section contains information on the current user's account. The most important for the performance are the following parameters:

- **Permissions** define the operations that the user can perform within the limits of his/her role(s).

Protection Studio interface. Services

Help menu

The **Help** menu is shown in figure below.



Information

contains general information about the current project and current user's

Project information

Information about the current project. Please pay special attention to the following information:

- **Protection version** defines the type of the product in use (FL Disc, FL ProActive, SF Crypto, etc.)
- **Expiration date** - the project becomes unavailable when this date comes.
- **Project users** - the list of users with access to the project, as well as their roles.
- **Project quotas** define the restrictions imposed on the total number of main operations for the product protection.

Account information

This section contains information on the current user's account. The most important for the performance are the following parameters:

- **Permissions** define the operations that the user can perform within the limits of his/her role(s).

Protection Studio interface. Help menu

Command	Purpose
Full user guide…	Switches to the product user guide in the Members Area of the StarForce website.
Quick user guide…	Displays PS quick user guide (select a section to view its help file).
Send request to customer support…	Sends an email to the product customer support: when selecting this command, a default e-mail client runs on the user's computer, and a message template is created.
About…	Displays the Protection Studio version and the **Send feedback** button (described above).

Help menu commands

3.2.1.2 Toolbar buttons

Button	Name	Purpose
🔄	**Reconnect…**	Returns to the connection window to select another protection server, workspace or project file. The Protection Studio window is closed and the connection window is displayed again.
📂	**Open**	Opens a project file (.PSF) saved to the user's computer.
💾	**Save**	Saves current project settings to a project file (.PSF).
💾	**Save as...**	Saves current settings to another project file (.PSF).
❓	**Help**	Displays Protection Studio Help.
ℹ	**About**	Displays the Protection Studio version.

Toolbar buttons

3.2.1.3 Using hot keys

Key combination	Command
Ctrl+R	**Reconnect**
Ctrl+O	**Open**

Key combination	Command
Ctrl+S	Save

Protection Studio interface. Hot keys

3.2.1.4 Working area

The main part of the Protection Studio window (see figure) is divided into two parts: the left tree-form panel contains a list of available working sections, while the right panel displays the contents of the selected section.

The sections are combined into two main groups: *informational* and *functional* groups.

Information group contains:

- the **Project** and **Account** sections. They display basic information about the current project (workspace) and the account. These data are specified when the project is created by the StarForce customer support engineers. The account parameters are the only data, which can be edited in the sections of this group.

Functional group contains:

- **General options** section – a parameter section that is mandatory and common for all services. Platform type, binding type, information about driver and key path in the registry are specified in this section. Information on read-only but important protection project parameters is also displayed there.
- **Files**, **Methods**, **Computer binding options**, **Protection GUI options** and **Advanced options** sections that constitute the **Settings** section. It is possible to specify most of the protection settings for the project there.
- **Protect files, SDK**, that represent the corresponding Protection Studio services. Each section has a description, data entry fields and service control keys. To run a service, it is sufficient to specify the options of the corresponding section and the **General options** section. Parameters for the protection of files can be specified in the **Settings** section.

3.2.1.5 Display of the protection process

There are two tabs in the lower part of the Protection Studio window: **Log** and **Errors** (see figure).

The **Log** tab contains information about the operation of all services, as well as information about the changes in the account parameters and custom rights when using SDK.

The **Errors** tab contains a list of warnings and errors (if any) detected during the launch of the last service.

The tabs close after one of the following operations is performed:

1. Clicking after the cursor is moved to a free field in the left panel or to any entry field in the right panel;
2. Double-clicking the caption of the corresponding tab.

Additional tabs with information about the operation of the services appear when running the services.

3.2.2 Specifying protection parameters

Mandatory parameters are distinguished from all the project information, since it is not possible to perform protection without these parameters. Mandatory parameters are described in this section. Other information is given in section **Complete_description_of_the_protection_project_settings** . Paths to the files for protection, disc label and the path to the output folder to save the protected files are the mandatory parameters when creating a protection project (see table in section "The scheme of implementing protection...").

After all the necessary data is added to the project, it is possible to switch to the protection process itself, by running the **Protect files** service (see **Protection of files**).

3.2.2.1 Files for protection

The main purpose of file protection is counteraction to the software product cracking, that is, to analyzing of its algorithms and source code.

The following files can be included into the list of files for protection:

- executable files (`.exe`) and dynamic link library files (`.dll`). Sometimes they are combined under the common name – program files.
- data files.

Protection of program files

When using this protection method, the program files are stored in the encrypted state. If the verification of running application succeeds, the files are loaded into the RAM and are decrypted in it. If verification fails, the protected application does not run.

It is possible to select the protection level during the protection implementation (see the **`Protection Level`** option in the **Advanced options** section in figure). Increasing the protection level improves the protection quality; however, it increases the size of the protection library. It should be remembered that increasing the protection level can sometimes lead to slowdown during the application operation.

Attention! Only unencrypted and uncompressed executable files can be protected. Code packing by UPX-type programs and encoding by other protection systems is meant here.

Data file protection

When protecting data files, their contents are encrypted and the files are placed and hidden in a container file. The contents of the file are decrypted automatically when accessed from the protected application.

The main point here is not the encryption of the file contents, but hiding the names of files by placing them into the container. If a filename is unknown, it is not possible to extract it from the container by any means. Therefore, the files located in the container cannot be found using the FindFirst/FindNext functions and the similar ones. If the filename is known, it is quite easy to extract the file from the container.

In the cases when it is not possible to hide the names of files located in the container, it is recommended to create dummy files or files with “garbage” with the same names as the names of files in the container. These files should be placed outside the container, so the protected application will read the files from the container and the files outside it will draw away cracker's attention. But it is important to note that this way of protection is quite weak and should be used rarely.

The main restriction to data files for protection is that they have to be read-only.

Since it takes little time to decrypt a data file from the container before referring to it, this method of protection can be applied to almost any data files, which are used by the application. The exception here is only files with critical access time.

The data files which are accessed by imaging them into the memory (File Mapping method) can be protected as well.

The data file protection is effective only in case if the names of these files are hidden; therefore:

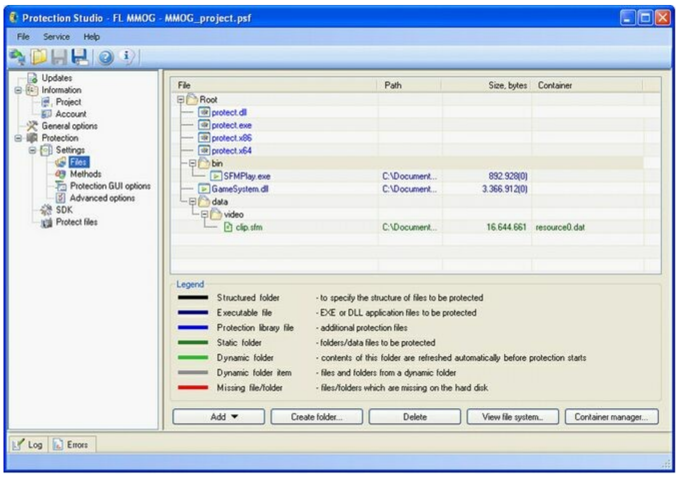
- The names of the protected files must not be listed in the protected application openly.
- The files should not be accessed every time the application runs.

The names of the files for protection should not be available in other versions of the application to be protected.

Data files are placed to the containers created by the user. After protection, all the files in a container are converted into a single file with the container name.

Attention! Container creation, as well as data file protection is only possible when protecting executable files and using the protection driver at the same time (see **Protection driver**). This option is enabled by the customer support when creating the protection project.

Adding files to the protection project is performed in the **Files** section (see below).



File	Path	Size, bytes	Container
Root			
protect.dll			
protect.exe			
protect.x86			
protect.x64			
bin			
SFMPlay.exe	C:\Document...	892.928(0)	
GameSystem.dll	C:\Document...	3.366.912(0)	
data			
video			
clip.sfm	C:\Document...	16.644.661	resource0.dat

Legend

▬	Structured folder	- to specify the structure of files to be protected
▬	Executable file	- EXE or DLL application files to be protected
▬	Protection library file	- additional protection files
▬	Static folder	- folders/data files to be protected
▬	Dynamic folder	- contents of this folder are refreshed automatically before protection starts
▬	Dynamic folder item	- files and folders from a dynamic folder
▬	Missing file/folder	- files/folders which are missing on the hard disk

'Files' section

When including files into the list, their paths are specified relative to the root directory of the software product to be protected, or relative to the folder with the protection library.

For detailed information and recommendations on protection of files see **'Files' section**.

Adding executable files to the protection project

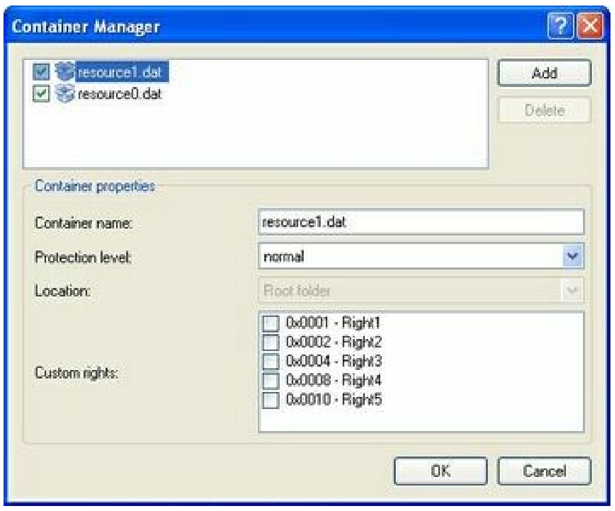
Attention! The names of the executable files from the protection project, containers, protection library and structured folders must contain only Latin letters, figures and admissible characters except for the following ones: * ? / \ | : < > " № The names of the structured folders, containers and protection library cannot be as follows: CON, PRN, AUX, CLOCK$, NUL, COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, LPT9, including any combinations of the CON.* type (for all the above-listed names irrespectively of the letter case).

1. Select a folder to add executable file(s) to.
2. Press **Add** (see figure above).
3. Select **Executable file(s)…** in the displayed contextual menu.
4. Select file (or several files while holding down the **Ctrl** key) in the displayed standard dialog box and press **OK**.
5. Selected files will be added to the list of executable files to be protected.

Adding data files to the protection project

Container creation

1. Press **Container manager…** in the **Files** section (see figure above).



☑ resource1.dat
☑ resource0.dat

Container properties

Container name: resource1.dat

Protection level: normal

Location: Root folder

Custom rights:
☐ 0x0001 - Right1
☐ 0x0002 - Right2
☐ 0x0004 - Right3
☐ 0x0008 - Right4
☐ 0x0010 - Right5

'Files' section. Container Manager window

2. Press **Add** in the displayed 'Container Manager' window (see figure) and specify the name of the container to be created.
3. Select the protection level for the container. **Normal** is the optimal level. Increasing the protection level increases the size of the container.
4. When limiting the rights for the containers, tick off the required custom rights.
5. Press **OK** to complete the operation.

Adding files

1. Press **Add** (see figure above).
2. Select **Data file to container…** in the contextual menu.
3. Select a file (or several files while holding down the **Ctrl** key) in the displayed standard

dialog box and press **OK**.

4. Select a container to add the file to. Please mind that a container is uniquely associated with the folder containing the object (a file to protect or a folder), which has been added to the container prior to the other objects. Later on, only the objects from this folder or from the enclosed folders can be added to the container for protection. The container is considered as *suitable* for such objects. Objects from the folders of higher levels can only be added to other containers.

Addition principle:

a) If not a single suitable container for a file has been created so far, specify a new container parameters; use the displayed ‘Container manager’ window to do so.

b) If there are several suitable containers for the file, select one of them in the displayed ‘Container selection’ window.

c) If there is only one suitable container, it is selected automatically.

5. Selected files will be added into the container.

Note. Usually, one container is used for protection. More than one container is used if the size of the first container is more than 2 GB (for example, when updating the protected product, see section **Update...**).

File deletion

1. Select a file to be deleted.
2. Perform one of the three operations:

   a) Press **Delete** button in the working area of the section;

   b) Right-click to open the contextual menu and select **Delete** in it;

   c) Press **Delete** key on the keyboard.

Notes

1. If one or several containers become empty after the objects for protection are deleted, the ‘Delete containers’ window appears that suggests deleting the containers. The containers are deleted by pressing **Delete**. If some containers must be kept, deselect their names.

   When pressing **Cancel**, the containers are not deleted and can be used again to place the objects for protection in the folders of any level in the working area.

Attention! There must be no empty containers when running the **Protect files** service: protection with empty containers is impossible.

2. It is possible to change the path to a file in the protection project. To do so, select a string with the filename so that a switch appears in the intersection of this line and the **`Path`** column. The switch opens a standard folder search window.

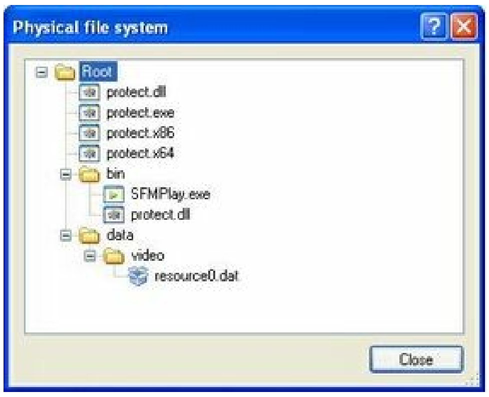
   If a folder without a file with the specified name is selected, all the symbols in this string are highlighted red.

3. It is possible to change the location of a data file. To do so, select a string with the filename; then a switch appears in the intersection of this line and the **`Container`** column. The switch opens a list of the containers; a container for the file can be selected

among the existing containers, or a new container can be created. To do so, seelct **`Container manager…`**, which results in displaying the 'Container Manager' window (see figure).

4. If the application to be protected has a structure, where an executable file and data files for protection are located in different folders relative to the root directory, it is necessary to form a chain of nested folders using **Create folder…** at first, and then select a folder of the required level and add files to the project.

   Thus, figure above indicates that the path of the executable file relative to the root directory is `\bin`, while the data file is in a container `\data\video\resource0.dat`. This structure can be viewed in the 'Physical file system' window, which opens when pressing **View file system…** button (see below).



Physical file system

3.2.2.2 Output folder

Files generated during the protection implementation (encoded files, protection library, data containers, etc.) are placed to the specified folder. This folder is the same as in the 'Options' window by default (see figure).

A folder on the computer to place the protected files can be redefined in the **Protect files** section in the **`Output folder`** field (see figure). The value of this field is saved to the project file.

The folder can be specified in one of the two ways:

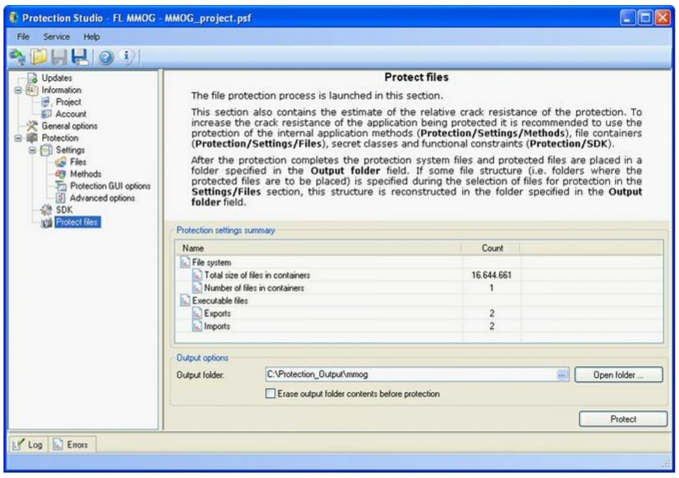
1. Explicit specification of the complete path and name of the folder;
2. Specification of the path relative to the project file location:

   a) `<folder name>` or `.\<folder name>` - the folder is placed beside the project file;

   b) `..\<folder name>` - the folder is placed one level higher than the project file;

   c) `..\..\ <folder name>` - the folder is placed two levels higher than the project file.

Attention! Using relative path is only possible if the project file has been saved at least once. Otherwise Protection Studio displays the following message when the service is run: "Folder with the specified path cannot be created!"



Protect files

The file protection process is launched in this section.

This section also contains the estimate of the relative crack resistance of the protection. To increase the crack resistance of the application being protected it is recommended to use the protection of the internal application methods (**Protection/Settings/Methods**), file containers (**Protection/Settings/Files**), secret classes and functional constraints (**Protection/SDK**).

After the protection completes the protection system files and protected files are placed in a folder specified in the **Output folder** field. If some file structure (i.e. folders where the protected files are to be placed) is specified during the selection of files for protection in the **Settings/Files** section, this structure is reconstructed in the folder specified in the **Output folder** field.

Protection settings summary

Name	Count
File system	
Total size of files in containers	16.644.661
Number of files in containers	1
Executable files	
Exports	2
Imports	2

Output options

Output folder: C:\Protection_Output\mmog

☐ Erase output folder contents before protection

‘Protect files’ section

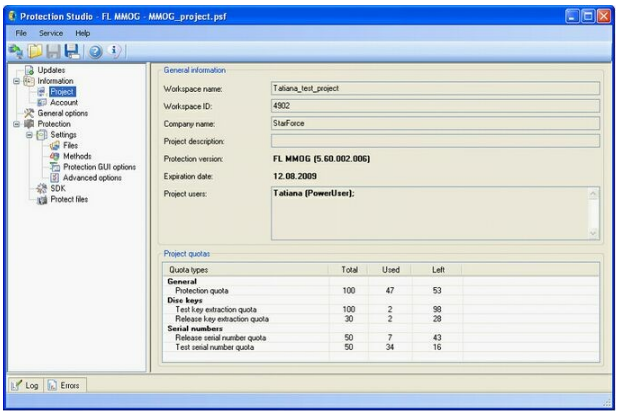
If it is necessary to delete the contents of the folder, **`Erase output folder contents before protection`** should be ticked off.

3.2.3 Complete description of the protection project settings

This section describes a protection project and its settings in detail. It contains information that was not included in the Specifying protection parameters section. Thus, these two sections contain exhaustive information about the possible settings in the protection project.

3.2.3.1 ‘Project’ section

Information about the protection project (workspace) is displayed in the **Project** section (see below).



General information

Workspace name: Tatiana_test_project

Workspace ID: 4902

Company name: StarForce

Project description:

Protection version: **FL MMOG (5.60.002.006)**

Expiration date: **12.08.2009**

Project users: **Tatiana (PowerUser);**

Project quotas

Quota types	Total	Used	Left
General			
Protection quota	100	47	53
Disc keys			
Test key extraction quota	100	2	98
Release key extraction quota	30	2	28
Serial numbers			
Release serial number quota	50	7	43
Test serial number quota	50	34	16

‘Project’ section

This information is specified by StarForce Customer Support during the project creation. It can be changed by StarForce Support only.

The section includes the following fields:

Field name	Contents
`Workspace name`	The name of the selected workspace where the protection by means of Protection Studio can be implemented at the moment. It usually coincides with the name of the product to be protected.

Field name	Contents
`Workspace ID`	The workspace ID in the StarForce system.
`Company name`	The name of the publisher/developer of the software product to be protected or the name of the protection project customer.
`Project description`	This field usually contains brief description of the project purpose; however, any information can be given for a user's convenience when working with various projects.
`Protection version`	The type of the product in use (FL MMOG in this case) and its version.
`Expiration date`	The project is not available after this date expires.
`Project users`	A list of users (and their roles) with access to the project.
`Project quotas`	The restrictions on the total number of main operations of the product protection, the restrictions on key extraction and Serial Number generation.

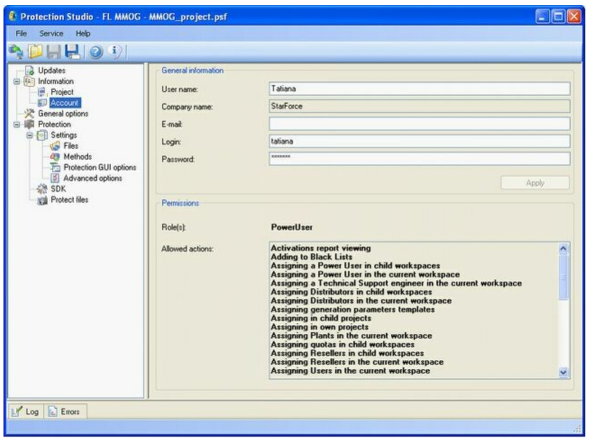
Fields in the 'Project' section

Quotas are the restrictions on the number of certain actions aimed at product protection (protection of files, etc).

Quotas are assigned to workspaces.

3.2.3.2 'Account' section

The **Account** section displays information about the user and his permissions (see below).


Protection Studio - FL MMOG - MMOG_project.psf

General information

User name: Tatiana
Company name: StarForce
E-mail:
Login: tatiana
Password: *******

Permissions

Role(s): PowerUser
Allowed actions:
Activations report viewing
Adding to Black Lists
Assigning a Power User in child workspaces
Assigning a Power User in the current workspace
Assigning a Technical Support engineer in the current workspace
Assigning Distributors in child workspaces
Assigning Distributors in the current workspace
Assigning generation parameters templates
Assigning in child projects
Assigning in own projects
Assigning Plants in the current workspace
Assigning quotas in child workspaces
Assigning Resellers in child workspaces
Assigning Resellers in the current workspace
Assigning Users in the current workspace

'Account' section

The section includes the following fields:

Field name	Contents
`User name`	Usually the user's first and last names.
`Company name`	The name of the company that the user represents.
`E-mail`	User's email – to obtain information about the project expiration and subscription messages.
`Login`	Login used to access the system.
`Password`	Password used to access the system.
`Role(s)`	Names of roles assigned to the user.

Field name	Contents
`Allowed actions`	A list of actions the user can perform in this workspace within the limits of his role(s).

Fields in the 'Account' section

Assigning an account to a workspace means that the corresponding person gains access to the workspace and its quotas. The main attribute of the account assignment is **role**.

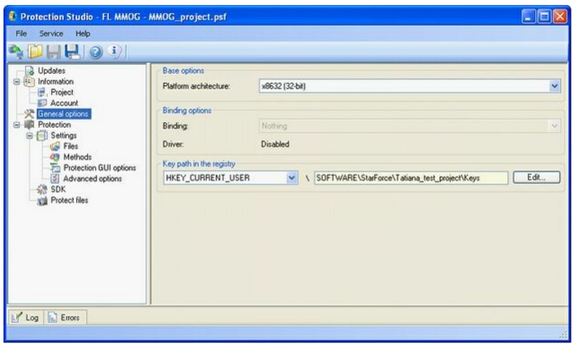
Role is a set of static permissions defining an aggregate of possible actions of a system user in a certain workspace. One user can have any number of unique roles in one workspace. In this case the permissions of all roles are summed up.

A user does not have access to role assignments for certain accounts; however, he/she can contact StarForce customer support if there is a need to restrict or extend access permissions of users, assigned to a workspace.

A user can assign user name, email, login and password at his discretion. In order for the changes to take effect, it is necessary to press the **Apply** button which becomes enabled when any changes are made. It should be kept in mind that new login and password should be entered when connecting to Protection Studio.

3.2.3.3 'General options' section

Besides the data specified by the customer support, the **General option** section contains parameters that can be modified (see below).


Protection Studio - FL MMOG - MMOG_project.psf

Base options
Platform architecture: x8632 (32-bit)

Binding options
Binding: Nothing
Driver: Disabled

Key path in the registry
HKEY_CURRENT_USER \ SOFTWARE\StarForce\Tatiana_test_project\Keys

'General options' section

Field	Possible values	Purpose
`Platform architecture`	• x8632 • x8664	Selecting the type of the product hardware platform: 32-bit or 64-bit.
`Binding`	• Nothing	This option is set by the customer support and cannot be changed.
`Driver`	• Disabled	The driver provides more reliable protection and allows data file protection. This parameter is set to **Disabled** by default. Enabling this parameter can be requested from the Customer support.
`Key path in the registry` (combo box)	• `HKEY_LOCAL_MACHINE` • `HKEY_CURRENT_USER`	Selecting a system registry section to store protection keys and data of the product. It is not recommended to use `HKEY_LOCAL_MACHINE`.

Field	Possible values	Purpose
`Key path in the registry` (edit box)	Characters allowed for system registry keys (Latin letters, upper and lower case, figures, \ ( ) { } [ ] '.', ':', underline, space, hyphen)	Defining an exact key path in the selected registry section.

Fields in the 'General options' section

Protection data in the system registry

Keys, license parameters and protection settings necessary to run the protected software are saved in the Windows system registry during first product launch (or during further runs, which takes place rarely).

The selection of the main registry section for storing the protection settings is defined by the type of the application to be protected. *It is recommended to use the* `HKEY_CURRENT_USER` *section for all types of applications except services*.

It is recommended to use the `HKEY_LOCAL_MACHINE` *section for the services to be protected*.

Such restrictions are imposed because the application has to be run with administrator rights in order to write data to `HKEY_LOCAL_MACHINE`, but an end user of the protected product does not always have such rights. There are also a number of restrictions when working with this section in Windows Vista operating systems.

If the protected application is run (for instance, for testing purposes) on the same computer the Protection Studio is installed on, then the **Edit...** button can be used to remove the protection settings from the registry (see also **Redefining the protection system settings**).

3.2.3.4 'Files' section

This section is a part of the **Settings** section.

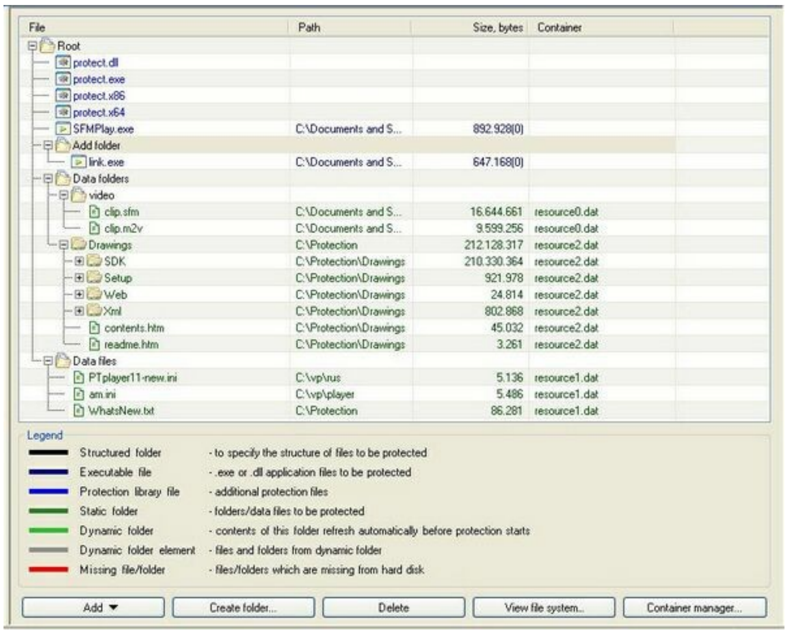
Files for protection are selected in this section (see figure) (see also **Files for protection**).

The section is a tree of the protection files (protect.dll, protect.exe, protect.x86, protect.x64), executable files, data files and folders with data files to be protected. For an executable file the following data are indicated: its size, path and overlay size (in brackets). For a data file – its size, path and a container the file is to be placed to during the protection. For a folder with data files and for a data file – the folder/file size, folder/file path and a container to place the folder/file to during the protection. If an executable file has no overlay, there is 0 value in the brackets.

The objects of different types are highlighted in different colors described in the **`Legend`** field.

If the name of a folder for a file in the section should be changed, it is necessary to select the file at first, then invoke a standard folder selection window, using a switch now available in the 'Path' column, and specify a new path for the file.

Files that require the same folder names or the same container can be selected simultaneously by holding down **Shift** key to select a group of successive elements, or **Ctrl** key to select separate items. In this case it is sufficient to specify a path or a container for one of the selected files, which applies to the whole group.



File	Path	Size, bytes	Container
Root			
protect.dll			
protect.exe			
protect.x86			
protect.x64			
SFMPlay.exe	C:\Documents and S...	892.928(0)	
Add folder			
link.exe	C:\Documents and S...	647.168(0)	
Data folders			
video			
clip.sfm	C:\Documents and S...	16.644.661	resource0.dat
clip.m2v	C:\Documents and S...	9.599.256	resource0.dat
Drawings	C:\Protection	212.128.317	resource2.dat
SDK	C:\Protection\Drawings	210.330.364	resource2.dat
Setup	C:\Protection\Drawings	921.978	resource2.dat
Web	C:\Protection\Drawings	24.814	resource2.dat
Xml	C:\Protection\Drawings	802.868	resource2.dat
contents.htm	C:\Protection\Drawings	45.032	resource2.dat
readme.htm	C:\Protection\Drawings	3.261	resource2.dat
Data files			
PTplayer11-new.ini	C:\vp\rus	5.136	resource1.dat
am.ini	C:\vp\player	5.486	resource1.dat
WhatsNew.txt	C:\Protection	86.281	resource1.dat

Legend

▬	Structured folder	- to specify the structure of files to be protected
▬	Executable file	- .exe or .dll application files to be protected
▬	Protection library file	- additional protection files
▬	Static folder	- folders/data files to be protected
▬	Dynamic folder	- contents of this folder refresh automatically before protection starts
▬	Dynamic folder element	- files and folders from dynamic folder
▬	Missing file/folder	- files/folders which are missing from hard disk

'Files' section

Buttons

Controls in the **Files** section (see above) are the buttons at the lower part of the work field and the corresponding contextual menu.

The Add button

It is possible to add (see above):

- an executable file to a selected (or created) folder.
- data files to a selected (or created) folder and to a selected container, or to a folder for protection inside a container.
- a folder with all or the most of the files to be protected to a selected (or created) folder and to a selected container; or to a folder for protection inside a container. The folder is added while preserving the file structure and all the subdirectories.
- a dynamic folder, which contents update automatically before the protection starts – to a selected (or created) folder and to a selected container, or to a folder for protection inside a container.

See also section **Files for protection**.

The Create folder… button

The folders are created (see above) to specify the relative paths for the files to be protected. The structure of the files to be protected relative to the `Root` folder is thus determined. The 'Folder properties' window, which is displayed when pressing **Create folder…** button, is used to create folders. A chain of nested folders can be created in one of two ways:

1. Create a folder. Then select it and proceed with the creation of a next level folder.
2. Specify the whole chain in the folder properties window by entering the path.

The Delete button

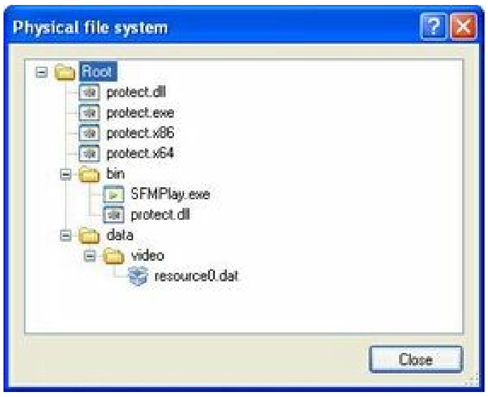
The button becomes enabled when an object for deletion is selected. Protection library files

(PROTECT.DLL, PROTECT.EXE or PROTECT.X86/X64 by default) and files/folders from the dynamic folders cannot be deleted.

The View file system… button

Pressing this button allows viewing the file structure of the protected files in the application being protected.

The window shown below displays the structure of the protected executable files and containers relative to a folder, specified in the **`Output folder`** field of the **Protect files** section (see figure) and indicated as `Root` in the **Files** section (see figure). The installation package of the protected application contains the PROTECT.DLL protection library file in each folder with an executable module, which is necessarily displayed in the 'Physical file system window'.



File system

The Container manager… button (see figure)

The 'Container Manager' window, which is being displayed when pressing this button, allows creating a container to add files and folders with data files to. Moreover, this window allows setting the container protection level and customer rights to it.

Contextual menu commands

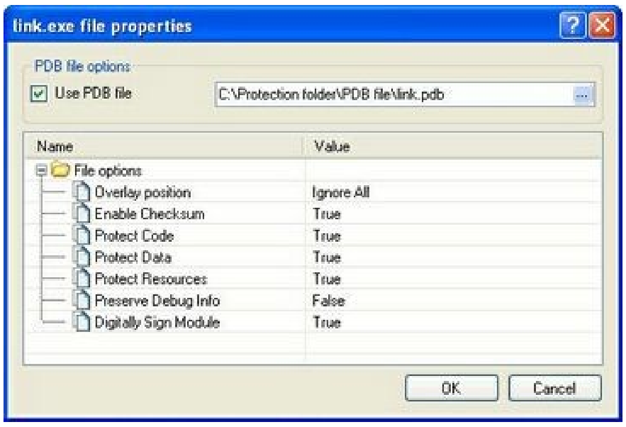
A contextual menu is displayed when right-clicking the work field. The appearance of the menu depends on the selected element (if any).

Command	Purpose
Add	Adds the following items to the selected element: • an executable file to the structured folder; • a file to a container or a static folder; • static or dynamic folder with files to a container; performing this command is similar to pressing **Add**.
Create folder…	Creates a structured folder inside a selected element: • Root folder; • structured folder; performing this command is similar to pressing **Create folder…**.
Delete	Deletes a selected element: • a file that is neither a dynamic folder file nor a protection library file (PROTECT. DLL, PROTECT.EXE or PROTECT.X86/X64 by default); • a folder that is not in a dynamic folder and does not contain protection library files; performing this command is similar to pressing **Delete**.

Command	Purpose
Container manager…	Opens the **Container Manager** window (see figure) to create a container; performing this command is similar to pressing **Container manager…**
View file system...	Displays the file structure of the protected files in the application to be protected; performing this command is similar to pressing **View file system….**
Expand all	Expands the file and folder tree in the work field of the section.
Collapse all	Collapses the file and folder tree in the work field of the section.
Properties	Opens a 'properties' window for the selected object: • the 'Protection library properties' window – for a protection library file (PROTECT.DLL, PROTECT.EXE or PROTECT.X86/X64 by default); • the 'File properties' window – for an executable file that is not in a container. This window is also displayed when one of the above-mentioned files is selected and double-clicked.
Rename	Opens a window to specify the name of the structured folder – the 'Folder properties' window.

'Files' section. Contextual menu commands

Work field element properties



☑ Use PDB file: C:\Protection folder\PDB file\link.pdb

Name	Value
File options	
Overlay position	Ignore All
Enable Checksum	True
Protect Code	True
Protect Data	True
Protect Resources	True
Preserve Debug Info	False
Digitally Sign Module	True

File properties window

1. The 'File properties' window is shown in figure above.

   The PDB file options field allows adding the debugging information from a `PDB` file to the project, in order to use this information during the protection implementation. The `PDB` file and the executable module should have the same name. It should be kept in mind that the `PDB` file parsing can take much time.

   If there is no `PDB` file when the **Protect files** service is run, although it is defined in the project, the system displays an error message and the service does not run.

   When a project file containing a `PDB` file is opened, the system checks if there were any changes not only for an executable module, but also for the corresponding `PDB` file.The **`File options`** table allows redefining the options related to the executable file protection quality.

Parameter	Possible value	Purpose
`Overlay position`	Ignore All;	Defines a way to process the overlays

Parameter	Possible value	Purpose
	Preserve Tail Data; Preserve Tail Data At The Same Position	during the file protection: - overlays are not preserved; - overlays are preserved, but their position relative to the beginning of the file may not coincide with the initial one; - overlays and their positions in the file are preserved (recommended).
Overlay is the additional data at the end of an executable module; these data do not suit the executable file format. The overlays are usually created by archivers and installers. Protection Studio detects the absence or the presence of an overlay and sets the **`Ignore All`** default value in the former case, while it sets the **`Preserve Tail Data At The Same Position`** value in latter case. The user can select another value from the list of possible value.		
`Enable Checksum`	True False	A check during the application launch: whether there were changes in the protected file.
`Protect Code`	True False	Encodes the .code section of the executable file.
`Protect Data`	True False	Encodes the .data section of the executable file.
`Protect Resources`	True False	Encodes the .resources section of the executable file.
`Preserve Debug Info`	True False	Determines whether a link to PDB file is deleted from the module.
`Digitally Sign Module`	True False	Adds a digital signature to the module. **Note**. The default value for this parameter is True for modules that have been signed prior to protection, while it is False for modules without the digital signature.

File properties

2. The ‘Protection library properties’ window allows redefining the name of the protection libraries (`protect` by default).

Attention! The name of the protection libraries must contain only Latin letters, figures and special characters, except for the following: * ? / \ | : < > " № `

3. The ‘Folder properties’ window allows renaming a structured folder.

Rebuilding the structure of the protected files

If the files selected for protection should be located in different folders in the protected software product, i.e. should form some file structure, the protected files should contain information about this structure. In order to do so, it is necessary to specify paths of the output (protected) files when forming the list of files for protection. The paths should be specified relative to the `Root` folder.

Since all the protected files and files of the protection library are loaded into the folder specified by the user (see Output_folder), after the protection completes, the loading is performed in the following manner if the paths of the output files have been specified:

- When specifying executable file paths relative to the root directory, the folders specified in the paths of the these files and containers are created automatically in the output folder, and the protected files are loaded into the folders according to these paths.
- For separate protected data files, path information is saved inside the containers.

In order to protect data files properly, it is necessary to keep their original file structure that means reconstructing this structure in a list of files for protection. So it is necessary to know how an absolute path to a data file inside a container is built:

Absolute path to a protected file inside a container is a path to the `Root` directory **+** path from the `Root` directory to the container, i.e. to the first directory in the hierarchy, where a file or a folder placed to the container is located, **+** path to the file inside the container (hierarchy of subdirectories).

The main way to set the paths inside a container is to add a folder with files to the container.

Adding a folder with files to a container

1. Select the required folder.
2. Press **Add** (see figure).
3. Select **Folder to container…** in contextual menu.
4. A standard dialog box appears; select a folder with most of the files to be protected and press **OK**.
5. If several containers have been created, select a container in the displayed ‘Container selection’ window. If there are no containers, specify a name for the container to be created.
6. Delete files that are not to be protected from the list.

The main way to set a path to a container is to create necessary folder tree in the `Root` directory, where a folder or files to be placed to the container are added to the lower level of the tree. The folder tree is formed by creating nested folders.

Folder creation

1. Select an object (`Root` or a structured folder), where a folder is to be created, and press **Create folder…**.
2. The ‘Folder properties’ window with an edit box to enter the folder name is displayed. After the name is typed in and **OK** is pressed, a folder with this name appears inside the selected object.
3. To relocate a file from the list to another folder, left-click it and move it to the required folder while holding down the mouse button (‘*drag and drop*’).

Recommendations for file protection

1. The applications written in the following programming languages are effectively protected:
   - C/C++ for Visual Studio, Builder and GCC;

- Delphi;
- Pascal;
- Visual Basic 6.0 with the exception of export protection;
- Visual Basic later than 6.0;
- Microsoft .NET.

2. For .Net applications, managed dlls are loaded when their code is used. Accordingly, protect.dll is loaded only when functions from a protected module are called, and the protection interface is displayed only after the module loading. Thus an unprotected application that uses a protected dll can operate until it calls a function from the protected module. To avoid this, the unprotected application should call a function for the protect. dll loading (e.g. LoadLibrary) when it starts operating. Another way is to protect the executable module along with dll.
3. When selecting files to be protected and the protection level it should be kept in mind that the excessive protection of some product components may lead to unacceptable loss of the application performance. The protection parameters should be modified according to the results of the protected product testing.

Recommendations for game protection

1. Many games allow the users to create their own modifications, alter game mechanics (gameplay) according to their needs, use new textures, make new maps, etc. – i.e. to perform modding.

   However, protecting the resources that can be modified prevents the users from doing so.

   Therefore, if the game architecture supports the creation of sequels or modifications, or it is supposed to issue some utilities to create maps, skins and modes after the games is released, it is not recommended to protect resources for such games.
2. After a demo version has been released, the following is recommended to prepare the release version of the application.
   - make the application modules (of demo and release versions) incompatible in the exports tables (see section below);
   - declare and protect additional functions (see section below);
   - modify the format of some data files, for example, levels.

3.2.3.5 ‘Methods’ section

This section is intended for selecting the executable file procedures (functions) to be protected.

Protection of functions is optional and is intended to significantly increase the level of the executable file code protection against cracking.Protection of functions includes two methods:

- Protection of internal (exported) functions of an executable file;
- Protection of method import tables, known as *protection of imports*.

Protection of internal functions (exports)

How it works

Protected internal functions are the *most important element* of the executable module protection in the **StarForce** system.

A protected internal function is moved from the source code of the developer's module into the protection core. When the application is executed and a call to the protected function is performed, the application transfers the control to the protection core and executes the function latently. The function should be placed into the table of exported functions in the source code, or a `PDB` file should be provided for the executable module, in order for the protection system to determine its entry point.

Types of protected functions

Functions to be protected can be divided into three groups:

Loopback functions. Besides the removal of a function from the module, the link to the function is removed from the table of exports of the module. Protection of a function as a loopback is the most recommended type of protection.

Public functions. The difference between these functions and the loopback functions is that the link to the protected function of this type is not removed from the table of exports. This protection method is applied to a function if it is also used by other application modules, i.e. it is a real exported function.

Callback functions. The calls to callback functions are performed by the protection core after the protected module is decrypted, but prior to its execution. Thus, a call to a callback function is hidden and is not present in the protected module. This method of a function protection is the most resistant to cracking.

Declaring exported functions

Using a special code

In order to declare a function as an exported one, the source project should include special code.

- When programming on C/C++, the `__declspec(dllexport)` keyword should be used. Example:

`__declspec(dllexport)` < function type> <function name>

`__declspec(dllexport) void InitAVICodec(void);` - for a callback function;

`__declspec(dllexport) int RunMasterMenu(CMenu &, int );` - for a loopback function.

- When programming on PASCAL, the `exports` keyword is used. Example:

```
exports
    YourFirstExportedFunction name 'YourFirstExportedFunction';
    YourSecondExportedFunction name 'YourSecondExportedFunction';
    YourThirdExportedFunction name 'YourThirdExportedFunction';
```

Attention!

The project should be compiled with the enabled linker option of generating fixups (relocation table.

- To enable this option in MS Visual C++ environment, **/FIXED:NO** should be added to the list of the project settings located on the **Link** tab of **Project Options** dialog.
- In Visual Studio.NET environment, this option is enabled in the **Additional Options** field of the **Linker** -> **Command Line** section of the 'project properties' dialog.
- In Borland Builder / Delphi environment, the compiler generates fixups by default. If compulsory generation of fixups is required, it is necessary to add the **/b:xxxx** linker option (Command-line switch = /b:xxxx). The 00400000h address is usually specified for Win32 .EXE files, the 10000000h address - for Win32 DLL files.

After the compilation completes, it is necessary to make sure the '.reloc' string is present in the protected module.

Using prefixes

There is a possibility to set the function parameters for their automatic application by the protection system. These parameters are formed as a prefix, which is concatenated to the front of the function name. If a prefix is specified, the function with the prefix is protected automatically using the specified method, as well as the execution speed (if specified) during the executable file protection. Thus, it is not necessary to set the function protection parameters manually in Protection Studio – it is more convenient to use prefixes to automate the process of the protection development. There are three types of prefixes that correspond to the three types of functions being protected:

- for callback functions: SFINIT<k>_<v>;
- for loopback functions: SFLB_<v>;
- for public functions: SFPROT_<v>;

where <k> is the number in the order of a callback function call;
<v> is the function execution speed (can take the values from 0 to 4, which corresponds to values from **`very slow`** to **`very fast`**, respectively; see table).

Example: SFPROT_3foo, where SFPROT is prefix, 3 – function execution speed, foo – function name.

Using PDB file

It is possible to protect internal functions without declaring them as exported ones. To do so, a PDB file that contains the debugging information and corresponds to the executable module should be added to the protection project in the **Files** section of Protection Studio (see 'Files' section). Consequently, it enables setting the protection parameters for all the internal functions of the file: the table of exports (the 'Exports' node) is replaced by the table of all its internal functions (the 'Internal functions' node).

At the same time, settings for the functions exported by the ordinal are removed. Protection parameters for internal functions are set just as for the exported functions.

If the **`Use PDB file`** field is deselected in the 'File properties' window of the **Files** section, the 'Internal functions' node is changed to the 'Exports' node; accordingly, only exported functions are displayed, while the settings for non-exported functions or functions exported by the ordinal, are deleted.

Field of application

The field of application of the protected functions is restricted by the delays during the execution. The delays depend on the protection profile, the protected function parameters, and the binding type. If the function has loops, its execution time can increase.

Attention! Run-time of complex functions can increase significantly after protection. To avoid slowdown during the protected application operation, StarForce recommends its customers to select functions to be protected very carefully and contact Customer support for advice.

Besides this, the following should be taken into account when protecting functions:

- The protection of functions is based on the hiding of the code. Therefore, it is unacceptable to distribute demo and other versions of the programs, which contain these functions in an unprotected state.
- The callback functions are the most effective in terms of protection; however, since a callback function can be called only once prior to the protected module startup, it is very difficult to select an appropriate callback function. Therefore, loopback functions should be given special attention.
- A function should not just initialize variables with constants. Otherwise, the results of the function execution would be enough for a hacker to restore the function.
- Functions to be protected as callbacks must not use functions from RTL of the compiler.
- It is possible to obtain good protection level by protecting about 10-20 exported functions that are crucial for the program operation. The size of each function should be several hundred bytes.

Note. It is not recommended to protect more than 20-25 export functions. Exceeding this value can result in shortage of the protection server resources and failure to process the query. The “Exception of type 'SystemOutOfMemoryException' was thrown” message is displayed in this case.

Attention! When protecting functions, it is necessary to remember that some functions are automatically transformed into inline by the compiler if their code is quite small. Since the protection of inline functions is impossible, all the functions to be protected should be defined as noinline in the source code of the application.

Example for Visual C++

```
__declspec( noinline )  void MyFunction()
{
  ...
}
```

Effectiveness

The protection of internal functions is the most effective method of protecting a program against cracking.

Protection of imports

How it works

Executable modules can call functions from other modules using the *dynamic linking*. Almost any executable module in Windows performs calls to functions defined in various dynamic link libraries (DLL).

Dynamic linking is implemented using methods of function *exporting* and *importing*.

When exporting functions, a table containing the names of the exported functions and their entry points is registered in the module. When importing functions, the following parameters are specified: the name of the module the function is exported from, the function name and the memory cell for the loader to place the function address to. When the protected module is run, the loader places it into the memory and searches the table of imports. If the table of imports contains a record for the name of the module, which is not loaded into the context of the process, the loader loads the module and searches for the required function in its export table. If a function is successfully located, the address of the function is placed into a corresponding record in the table of imports. Otherwise, an error message appears.

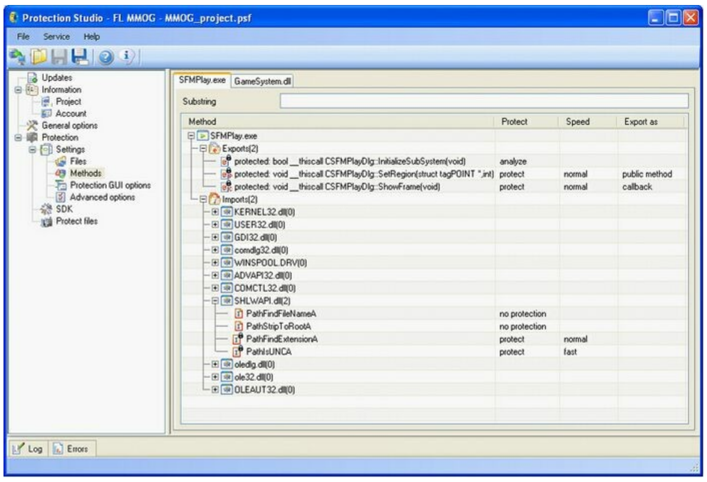
The process of restoring the executable module using the memory dump is rather complicated without knowing which imported function is called for the given cell of the table of imports. Therein the main idea of protecting the calls to imported functions consists in. In order to hide the calls to the module imported functions, StarForce protection removes its table of imports and restores it after the verification of the running copy. Some of the calls to imports are replaced with the calls to the protection core, which provides the call hiding.

Field of application

There is one significant limitation for the protection of calls to imports: the time between the transfer of control to the protection core and the call to the protected function should be sufficiently long to reliably hide the calls. This requires careful consideration when selecting imported functions to protect. It is recommended that an imported function with a protected call should not be called with frequency exceeding 20 times per second.

Efficiency

The method is only effective if the number of the protected calls to imported functions is big enough. The minimum number of the protected imports is 10-15 functions per module.



Method	Protect	Speed	Export as
SFMPlay.exe			
Exports(2)			
protected: bool __thiscall CSFMPlayDlg::InitializeSubSystem(void)	analyze		
protected: void __thiscall CSFMPlayDlg::SetRegion(struct tagPOINT *,int)	protect	normal	public method
protected: void __thiscall CSFMPlayDlg::ShowFrame(void)	protect	normal	callback
Imports(2)			
KERNEL32.dll(0)			
USER32.dll(0)			
GDI32.dll(0)			
comdlg32.dll(0)			
WINSPOOL.DRV(0)			
ADVAPI32.dll(0)			
COMCTL32.dll(0)			
SHLWAPI.dll(2)			
PathFindFileNameA	no protection		
PathStripToRootA	no protection		
PathFindExtensionA	protect	normal	
PathIsUNCA	protect	fast	
oledlg.dll(0)			
ole32.dll(0)			
OLEAUT32.dll(0)			

‘Methods’ section

The right panel of the **Methods** section consists of a set of tabs. Each tab represents a work field for each executable file from the list of files for protection. A tab caption is the name of the corresponding file.

For convenience of a function search, there is the **`Substring`** field (filter) to enter the name or a part of the name.

The work field is a table of exported (the ‘Exports’ node) and imported (the ‘Imports’ node) functions of a program file. Imported functions are grouped by libraries they are called from.

The total number of functions for protection in a node/library is specified in the brackets after the name of the node/library.

The table displays the following data for each function:

Column title	Possible value	Purpose
Method	any	Function name.
Protect	no protection analyze protect	Protection implementation method (for exports and imports, respectively).
	no protection protect	
Speed	very slow slow normal fast very fast	The required execution speed of the function (the faster the function is executed, the lower is the protection level, and vice versa).
Export as	public callback loopback	Function protection type (for exports only).

Function parameters

The **Speed** and **Protect** columns display information only if the **`protect`** value is selected in the **Protect** column.

It is possible to use the following types of the method protection for each program file.

Protection method	Performed by
Analyze a function	Protection server when protecting files. This method is applied to all exports by default. If this value is selected, the function is being analyzed and some internal transfers may be protected in it.
Protect a function	User. This method is recommended for a qualified user only. For imports, it is only possible to define the speed in this case.
Marking a function as not to be protected	User. This mode applies to all imports by default.

Possible ways to protect methods

Setting the protection parameters when selecting the **`protect`** value requires good knowledge of the product source code and is recommended for a qualified user only.

The order of the function protection

1. Select a function or several functions the protection parameters are to be set for (see figure above). Functions that require the same parameters can be selected simultaneously: press **Shift** key to select a group of successive functions, or **Ctrl** key to select separate items. In this case it is sufficient to specify the parameters for one of the selected functions, which applies to the whole group.

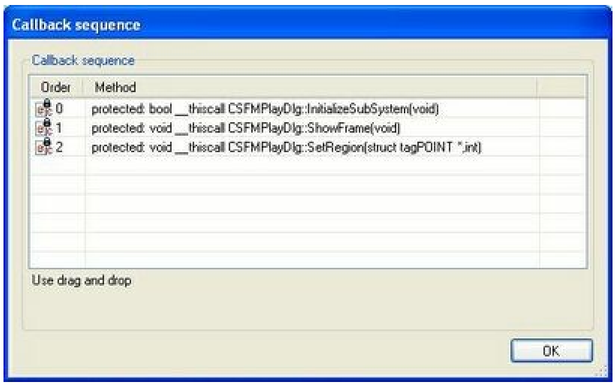
1. Select one of the three values in the dropdown list in the ‘Protect’ column:
   - **`analyze`** (default value, for exports only)
   - **`protect`**
   - **`no protection`** (default value for imports).
2. When selecting **`protect`**, select one of the five values in the dropdown list in the ‘Speed’ column:
   - **`very slow`**
   - **`slow`**
   - **`normal`**
   - **`fast`**
   - **`very fast`**.
3. When selecting **`protect`**, select one of the three values for the protection method in the ‘Export as’ column (***for exported functions only***):
   - **`Public method`**
   - **`loopback`**
   - **`callback`**.

Contextual menus of the work field

A contextual menu is displayed when right-clicking the work field. The appearance of the menu depends on the place in the table, where the menu is being opened.

1. The **Expand all** and **Collapse all** commands are displayed always and manage the representation of the function structure in the work field of the section by expanding and collapsing the trees of imported and exported functions.
2. The **Callback sequence…** command is displayed when the `callback` type is selected for two exports at least. Performing this command opens the ‘Callback sequence’ window (see figure).

   It is possible to change the callback sequence using *drag and drop* method, i.e. to left-click a function and move it to the required place while holding down the mouse button. The sequence defines the order in which the protected functions are called from the protection library.


Callback sequence

Order	Method
0	protected: bool __thiscall CSFMPlayDlg::InitializeSubSystem(void)
1	protected: void __thiscall CSFMPlayDlg::ShowFrame(void)
2	protected: void __thiscall CSFMPlayDlg::SetRegion(struct tagPOINT *,int)

Use drag and drop

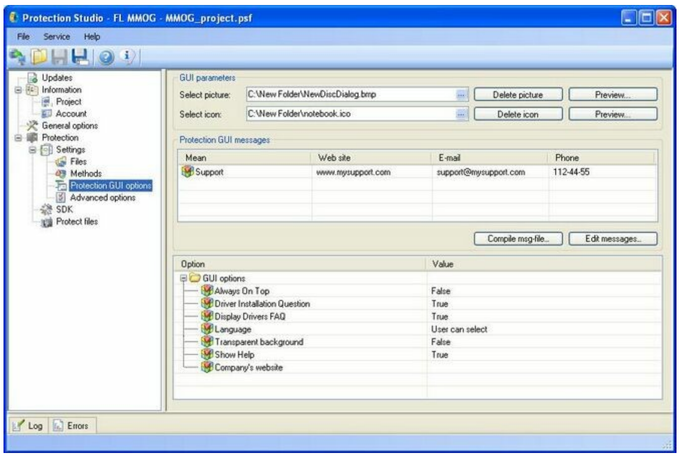
'Methods' section. 'Callback sequence' window

3. The **Properties…** command is displayed in the contextual menu, if an exported function with the **`protect`** value in the **Protect** column is selected. Performing this command opens the 'Method properties' window that allows setting the custom rights for the exported functions.

   It should be kept in mind that the custom rights specified for a function when the **`protect`** value is selected, are not saved if either the **`analyze`** method or '**`no protection`**' method is selected after that. If the **`protect`** value is selected again for this function, the required custom rights should be specified again in the 'Method properties' window.

3.2.3.6 'Protection GUI options' section

This section is used to define the three groups of parameters: GUI parameters, Protection GUI messages and additional GUI options (see below).



GUI parameters

Select picture: C:\New Folder\NewDiscDialog.bmp

Select icon: C:\New Folder\notebook.ico

Protection GUI messages

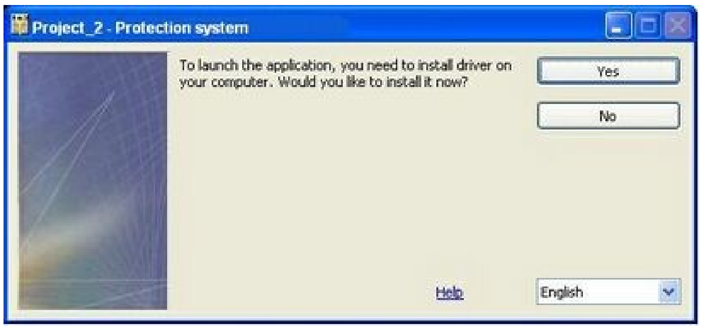
Mean	Web site	E-mail	Phone
Support	www.mysupport.com	support@mysupport.com	112-44-55

Option	Value
GUI options	
Always On Top	False
Driver Installation Question	True
Display Drivers FAQ	True
Language	User can select
Transparent background	False
Show Help	True
Company's website	

'Protection GUI options' section

- The GUI parameters group is intended for selecting a picture and an icon for the graphical interface; the picture and the icon can be subsequently viewed or deleted by pressing the corresponding buttons on the right-hand side.

  An example of the protected application interface is shown in figure below, where:
  - the icon is in the upper left part, in the caption (.ico format with following configuration: 16x16 and 32x32 pixels);
  - the picture is on the left-hand side of the window (.bmp, 113x195 pixels).



To launch the application, you need to install driver on your computer. Would you like to install it now?

Protection interface

- The Protection GUI messages group allows specifying contact details that are to be displayed in the protection messages, as well as editing the protection messages:
  - URL, email and phone number that are to be displayed to an end user can be specified in the table with contact details.

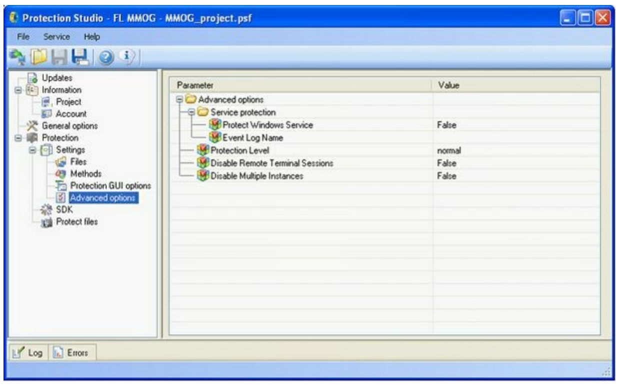
Mean	Purpose
`Support`	Information to contact the product customer support.

The contact details group for an end user

It is necessary to press **Edit messages…** to edit the protection messages. Pressing this button invokes Message Editor described in the **Message Editor** section

3.2.3.7 'Advanced options' section

This section is shown in figure below.



Parameter	Value
Advanced options	
Service protection	
Protect Windows Service	False
Event Log Name	
Protection Level	normal
Disable Remote Terminal Sessions	False
Disable Multiple Instances	False

'Advanced options' section

Advanced parameters values define the protection level and possible ways of application use.

Parameter	Possible value	Purpose
`Protection profile`	low protection level, high protection speed, small protection library size optimal high protection level, low protection speed, large protection library size	Selection of the encryption level. Increasing the encryption level improves the protection quality, but at the same time it increases the protection library and the run-time of the protected functions.

Parameter	Possible value	Purpose
`Disable remote terminal sessions`	False True	If enabled, it is impossible to run the protected application from server using terminal sessions (remote desktop connection). Thus, it prevents the protected application from being used by several users with the same license simultaneously.
`Disable multiple instances`	False True	Excludes a possibility to run more than one instance of the application on one computer.

Advanced protection parameters

Protection of services

These parameters are defined when applying protection to Windows services.

Windows service is an application, which can be run before a user logs in to the system. To implement services protection it is necessary to provide their operation without a user interface. The user obtains two products: the protected service and an activator, i.e. a tool intended for the license creation and activation. When installing the service, the user should run the activator and perform activation of the service using a standard interface of the protected software product. After that, the service will use the already activated license at its startup and if errors occur, the information will be written to Windows EventLog.

Since services do not display dialog boxes and all their messages are saved to the system event log, the parameters listed below define a manner of saving the protected service messages to the event log.

Parameter	Possible	Purpose
`Protect Windows services`	False True	Implementation of the service protection
`Event log name`	Name	Service name in the Event Log.

Service protection options

3.2.4 Protection of files

Protection of files is performed for each project after all the required parameters are set. File protection service is represented by the corresponding section shown in figure above.

The section contains two groups of fields.

The Protection settings summary group displays a table with information about the total number and size of files in containers, and the number of exported and imported functions selected for the protection.

The Output options group allows redefining the output folder (see **Output folder**).

This group also allows the deletion of the folder contents prior to the protection implementation by checking the corresponding checkbox.

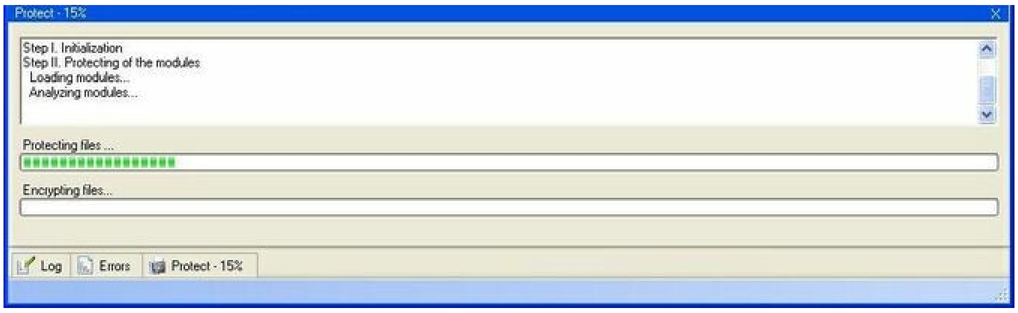
Attention! Folder contents cannot be restored.

The service is run by pressing the **Protect** button or **Enter** key.

If errors are detected in the project, an error message is displayed (see **Troubleshooting**

during the protection implementation), and the service terminates.

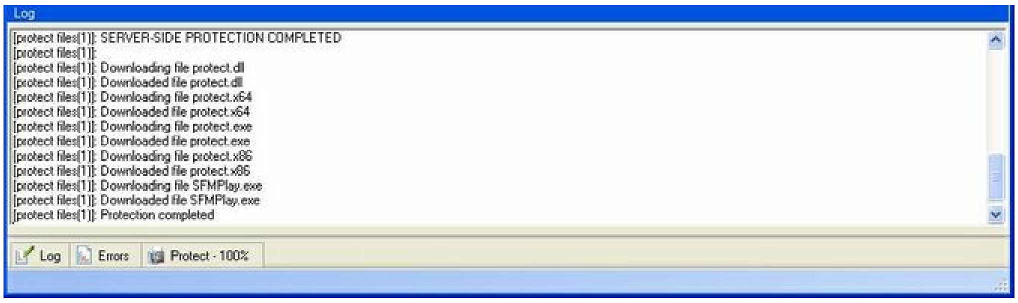
If no errors are detected, the protection process begins, and the accompanying messages are displayed in the **Protect** tab, while the progress of the protection is indicated by the progress bar and in percentage terms (see below).



```
Step I. Initialization
Step II. Protecting of the modules
  Loading modules...
  Analyzing modules...
```

‘Protect’ tab

Besides, all information about the operations being performed is displayed in the **Log** tab (see below).



```
[protect files(1)]: SERVER-SIDE PROTECTION COMPLETED
[protect files(1)]:
[protect files(1)]: Downloading file protect.dll
[protect files(1)]: Downloaded file protect.dll
[protect files(1)]: Downloading file protect.x64
[protect files(1)]: Downloaded file protect.x64
[protect files(1)]: Downloading file protect.exe
[protect files(1)]: Downloaded file protect.exe
[protect files(1)]: Downloading file protect.x86
[protect files(1)]: Downloaded file protect.x86
[protect files(1)]: Downloading file SFMPlay.exe
[protect files(1)]: Downloaded file SFMPlay.exe
[protect files(1)]: Protection completed
```

‘Log’ tab

After the protection completes, pressing the **Open folder…** button (see figure) opens the folder that is specified in the **Protect files** section (see **Output folder**) and contains the following files:

File types	Comments
Protected program files	The list of the protected files as defined in the **Files** section.
Container files that contain protected data files (if any)	
Protection library file (`PROTECT.DLL` by default)	All these files have the same name defined in the **Files** section (but different extensions).
Executable file of the protection interface (`PROTECT.EXE` by default)	
Protection installation files for different OS (`PROTECT.X64` and `PROTECT.X86`)	
File with protection interface messages	This file is generated on the basis of information entered in the Message Editor (see **Message Editor**) and only if such information is present in the project. The filename coincides with the protection library name. The extension is `.msg`. If no changes to the protection interface messages are made, no such file is generated.

Files generated by the ‘Protect files’ service

These files are placed to the folder according to the file structure in the **Files** section (see figure), while the `protect.dll` protection library file is placed to each folder containing a protected executable file.

Note! It is necessary to close the **Protect** tab (see above) before running the **Protect files** service once more; the **Protect** button becomes available only after the **Protect** tab is closed.

3.2.5 Creating the product installation package

Before testing the application it is necessary to form the installation package of the protected application.

The installation package has to include the following files.

1. Protected executable files of the application (`EXE` or `DLL`).
2. Protection library files (`PROTECT.DLL`, `PROTECT.EXE`, `PROTECT.X86/X64` by default), which should be placed according to the structure of the protected application. `PROTECT.DLL` should be placed to all the folders containing protected executable files.

Attention! It is not recommended to place the protected product in such a way that the protection library files are placed into a system folder.

3. Data container files (if any). Container files should be placed into folders according to the structure specified in the **Files** section.
4. A `GUI` file (`PROTECT.GUI` by default), if the graphical protection interface settings are saved to an external file (the file is generated using StarForce SDK, see **StarForce SDK**).
5. An `MSG` file (`PROTECT.MSG` by default); it is generated only if any changes to the standard message text were made (see **Message_Editor**). This file should be placed to the folder containing `PROTECT.EXE` and `PROTECT.X86/X64`.

All the aforementioned files are already in the folder specified in the **Protect files** section after the protection completes.

The installation package should also contain all other application files except for the protected executable modules and protected data files.

At the final protection stages, after the installation package is assembled, if a special installation program is created for the automatic application installation on a user's computer, the installer should be recompiled considering all the changes within the application installation package (replacing original files with the protected files and adding service files of the protection).

The installation script of the product may also contain operations on redefinition of the protection options by system registry keys (to provide the flexibility of the protected application settings). For the detailed description and a list of options available for redefinition see section **Redefining the protection system settings**.

3.2.6 Running services from the command line

Services can be run from the command line with the parameters that allow connecting to the protection servers without the use of the Protection Studio interface. Thus they can be run from the batch files and scripts, which allows partial automation of the application protection process.

Service name	Command line parameter	Description
Generate SDK	-Action:GenerateSDK	Runs the SDK generation service
Protect files	-Action:ProtectFiles	Runs the file protection service
Update	-Action:Update	Runs PS2 automatic update service

Services with the command line parameters

Parameters to run the services from the command line:

#	Name	Description
1	-Login:	StarForce account login.
2	-Password:	StarForce account password.
3	-Host:	Protection server.
4	-WorkspaceId:	Workspace ID (protection project).
5	-Project:	Full path and the name of the project file (PSF).
6	-Action:	An action (service) to be performed. The available actions are listed above.
7	-Log:	Full path and the name of the file to record the log.
8	-OutputFolder:	A folder to place all the files obtained during the service operation.
9	-FolderWithExeFiles:	A path to a folder that contains executable files (exe and dll) to be protected.

Parameters to run the services from the command line

Notes:

1. No inverted commas are required after a colon (":") (correct example is **`-Login:Roman`**).
2. Only two parameters (**`-Action:`** and **`-Log:`**) are mandatory in the general case.
3. If the **`Save connection settings`** checkbox in the 'PS connection' window is checked (see figure), the **`-Login:`**, **`-Password:`** and **`–Host:`** parameters should only be specified if these values should be redefined.
4. If the **`Next time use selected workspace automatically`** checkbox in the 'PS connection' window is checked (see figure), the **`-WorkspaceId:`** parameter should only be specified if this value should be redefined.
5. The order of the parameters is of no importance.

The return codes:

Code	Value
0	The process has completed successfully.

Code	Value
1	Command line error (if two mandatory parameters are absent, or incorrect path to the log or the project file is specified).
10	Account is incorrect.
11	Protection server is incorrect.
12	Incorrect workspace ID is indicated (it cannot be transformed into figures).
13	The system cannot obtain information about the workspace (probably, it does not exist or it is not assigned to the account).
14	The system cannot find a project file in the specified path.
15	Log file is not specified.
16	An error in the output folder name.
21	An operation the user selected does not exist, or it is not available for the user.
22	An error occurred during the service operation. The detailed information must be in log files.
23	An error occurred during the automatic update. The detailed information must be in log files.
24	The application is closed by the user (if the user closes Protection Studio launched from the command line without the parameters, or not from the command line).
25	The folder does not exist in the specified path.

The return codes

3.3 Testing the protected application

The protected product testing is a very important stage, since only by having the testing results it is possible to estimate the acceptability of the selected level of the software product protection, and the possibility to distribute the product among the end users.

Protected application operation must be tested in order to ensure that the applied protection does not affect the main performance capabilities of the application, i.e. it is necessary to make sure that the operation speed and the size of the protected application meet the specified requirements.

Therefore, the operations should be performed according to a typical scheme of new product testing, and it is necessary to make certain that all the functionality of the original (unprotected) application is fully supported in the protected version. That is why it is recommended to perform the full final testing of the product before and after the final protection of its files is performed.

3.4 StarForce SDK

3.4.1 SDK structure

StarForce SDK is intended to develop programs that are to be protected with the StarForce protection system and that use advanced StarForce features available via API.

StarForce SDK can be used to model the protected application behavior in different cases on the development and testing stages.

StarForce SDK also includes the secured classes, which are used to increase the level of the product protection against cracking.

SDK includes a package of files to be integrated into the application being protected:

- `protect.dll` – a debug version of the protection library. The functionality of the protection library included in the application package after the final protection remains the same in this version. Moreover, the debug protection library includes some additional checks and at the same time it allows skipping some slow initialization steps, such as disc check and activation. The debug version does not contain protection against debuggers as well, so there is a possibility to perform the full debugging of the protected application;
- `protect.lib` – a library of exports from the `protect.dll`, which is used to link the `protect.dll` and the application;
- `protect.exe` – the protected application GUI. This file is identical to the file generated during the final protection. In particular, it contains the same checks of the OS version, etc;
- `protect.x86` – to support Win32 platform for the x8632 processors,
- `protect.x64` – to support Win64 platform for the x8664 processors; in case of protection with the driver, both files can be present.
- PsaApi.h – a file for C/C++ with the declarations of the API functions that are used to obtain information about the licenses, as well as to declare the protected classes for C++ (if the latter are used).
- PscApi.h – a file for C/C++ with the declarations of the API functions used for the application initialization.
- PsConstants.h – a file for C/C++ with the description of constants used in SDK (the returned values of the API functions mainly).
- PsConfig.h – a configuration file for C/C++ containing the protection project settings. This file includes a set if `#define` instructions such as the path for the protection keys in the registry, etc.
- Files in the Delphi folder (PsaApi.pas, PscApi.h, PsConstants.pas, PsConfig.pas) are the analogues of the PsaApi.h, PscApi.h, PsConstants.h, PsConfig.h files to integrate SDK into the applications in Borland Delphi.

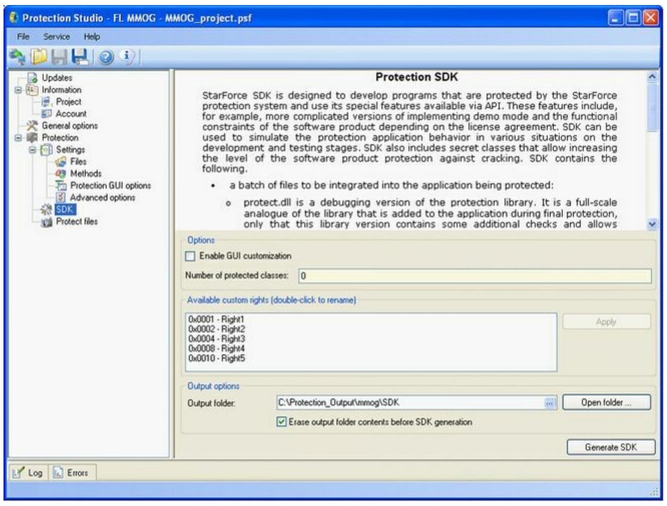
Note. Files with the `protect` name will in fact have the name defined in the project settings (see below).

`LIB`-file can be generated manually by creating your own DLL that has empty functions with the names coinciding with the names of the StarForce API functions. It may be necessary when applying a compiler that cannot use `protect.lib` from the SF SDK, for example, Microsoft Visual Studio 6.0.

Note. Some compilers (e.g. Visual C++ 7.1) do not support source code in Unicode. The PsConfig.h file is delivered in Unicode; therefore, this file should be converted into ANSI to work with Visual C++ 7.1. It is necessary to check that the product and company names specified in the file are correct (if they are not in Latin).

SDK files are generated individually for each protection project. The SDK package varies depending on the project settings.

3.4.2 SDK generation



Protection SDK

StarForce SDK is designed to develop programs that are protected by the StarForce protection system and use its special features available via API. These features include, for example, more complicated versions of implementing demo mode and the functional constraints of the software product depending on the license agreement. SDK can be used to simulate the protection application behavior in various situations on the development and testing stages. SDK also includes secret classes that allow increasing the level of the software product protection against cracking. SDK contains the following.

- a batch of files to be integrated into the application being protected:
  - protect.dll is a debugging version of the protection library. It is a full-scale analogue of the library that is added to the application during final protection, only that this library version contains some additional checks and allows

Options

☐ Enable GUI customization

Number of protected classes: 0

Available custom rights (double-click to rename)

0x0001 - Right1
0x0002 - Right2
0x0004 - Right3
0x0008 - Right4
0x0010 - Right5

Output options

Output folder: C:\Protection_Output\mmog\SDK

☑ Erase output folder contents before SDK generation

‘SDK’ service

To generate and obtain the SDK perform the following operations:

1. Load the required project file or create a new one.
2. Change the name of the protection library (when necessary) in the **Files** section (see figure).
3. Set the required parameters in the following sections: **Protection GUI options**, **Advanced parameters**.
4. In the **SDK** section (see above):
   a) Enable GUI customization (when necessary) by checking the corresponding checkbox;
   b) Define the number of the protected classes used by the protected product developers working with the StarForce SDK (in the **`Number of protected classes`** field). It is possible to specify not more than 4 classes; if the specified value exceeds 4, the entry field is highlighted red;
   c) Specify a folder to load the SDK files (in the **`Output folder`** field). It is also possible to delete the contents of the folder prior to the SDK generation by ticking off the corresponding checkbox. The contents cannot be restored.
   d) Press **Generate SDK** button or **Enter** key.

Note. Available custom rights can be renamed. To do so, it is necessary to double-click the required element in the corresponding field and enter a new name.

If there are errors in the project, an error message appears (see **Troubleshooting during the protection implementation**) and the service terminates.

If no errors are detected, the service begins operation.

During the service operation the information about the performed operations is displayed in the **Generate SDK** tab, while the progress of the process is indicated by the progress bar.

After the process completes, the corresponding message appears.

Pressing **Open folder…** opens the folder that is specified in the **`Output folder`** field and contains files to be integrated into the application being protected.

Note! It is necessary to close the **Generate SDK** tab before running the service once more; the **Generate SDK** button becomes available only after the tab is closed.

3.4.3 Applying secured classes

The library of secured classes is used for deeper integration of the protection into the application being protected. It provides protection of internal variables in order to prevent them from being changed and substituted. It also provides protection of the source code against disassembling and reverse engineering, which makes it an efficient method to counteract hacker attacks. The library of secured classes is a set of C++ classes, which are based on integrated integer data types and which are used in mathematical operations.

Secured classes are the classes that substitute integrated data types, while the values of the corresponding variables are encrypted and stored inside the secured class:

```
class PsUInt1
{
private:
unsigned int var1;
};
```

Currently the secured classes are supported only for one data type – unsigned int.

The number of secured classes is defined by the developer of the protection project, and their declarations are placed in the PsaApi.h (PscApi.h) file during the StarForce SDK generation.

Using the secured classes instead of ordinary integer types results in additional calls to the protection core, which complicates the reverse engineering significantly.

Attention! Using the secured classes in time-critical code sections can significantly slowdown the application, as the secured classes are called relatively slow (up to tens milliseconds per operation).

In order to embed the secured classes into the application prior to the StarForce SDK generation, specify the required number of secured classes in the corresponding field in the **SDK** section (see figure). It is 0 by default.

Following this, the variables of the unsigned int type can be replaced by the variables of secured classes.

Example:

Before the replacement

```
#include <stdio.h>
unsigned int MyFunction(unsigned int i, unsigned int j )
{
            return i * i + j * j + i * j;
}
```

After the replacement

```
#include <stdio.h>
#include <PsaApi.h>
unsigned PsUInt1 MyFunction(PsUInt1 i, PsUInt1 j )
{
            return i * i + j * j + i * j;
}
```

Replacement of the embedded data types with secured classes does not change the methods of a function call. For instance, the call of the `MyFunction(5, 3)` format is valid. All the required transformations are automatically inserted during the application compilation.

Attention! When compiling the application using the secured classes, it is necessary to include the substitution of inline functions; otherwise, the protection with the secured classes will not take effect. However, since the inline function substitution has to be disabled when protecting loopback functions, special attention should be given when placing #inline directives in the source code.

3.4.4 Protection API functions

`PSA_CheckProtectedModulesReadOnlyMem`

The function checks the integrity of code and data.

`PSA_FsOpenFile`, `PSA_FsCloseFile`, `PSA_FsGetFileSize`, `PSA_FsSetFilePosition`, `PSA_FsReadFile`

The functions read a data file, if there is the file in a container (in the SDK mode, the functions operate with the files on a hard disk, not in a container).

`PSA_FsVerifyFileSignature`

The function checks the integrity of the digital signature of a data file, if there is the file digital signature in a container. In the SDK mode, the function always returns PSC_STATUS_SUCCESS.

`PSA_CryptedTrafficOpen`, `PSA_CryptedTrafficEncrypt`, `PSA_CryptedTrafficDecrypt`, `PSA_CryptedTrafficClose`

The function encrypts traffic at the client end. The generated SDK also contains Java modules to implement traffic encryption on server.

`void __stdcall PSA_DummyFunction();`

This function has null functional and is designed to prevent optimization during the linking procedure and deletion of the link to protect.dll, if no other protection API functions are called at the moment.

```
unsigned __int32 __stdcall PSA_IsDemoMode ( bool *isDemoMode);
```

The function checks if the application launch in demo-mode takes place.

When using the simplified initialization mode (see **Modeling different scenarios of the protected software product behavior by means of SDK** below), the value is read from the registry, the key is `IsDemoMode`, the key type is REG_DWORD.

```
unsigned __int32 __stdcall PSA_GetFeaturesGrantedByLicense
( unsigned __int32 *features );
```

The function defines a set of options allowed for the current license, as a bit mask (5 bit, [PSA_GrantedFeature0 … PSA_GrantedFeature]).

When using the simplified initialization mode, the value is read from the registry, the key is FeatureSetGrantedByLicense, the key type is REG_DWORD.

```
unsigned __int32 __stdcall PSA_CheckFeaturesGrantedByLicense
( unsigned __int32 features );
```

A debugging function; it is usually ignored during protection.

It is designed to debug functions with access limited by the license type. It compares the bit mask of the current license features with certain requirements (i.e. it emulates a check performed by the protection system during the execution of a function with access restriction depending on the license type), and returns an error message if none of the bits in the masks of the license features and function requirements coincide.

```
unsigned __int32 __stdcall PSA_GetLicenseStoragePath ( wchar_t
*pathBuffer, size_t *pathBufferSizeInWideChars, HANDLE
*registryBaseHandle );
```

The function returns the path to the licenses in the registry.

```
unsigned __int32 __stdcall PSA_GetLicenseLifeTimeLimit ( unsigned
__int64 *licenseLifeTimeLimit );
```

The function returns the license lifetime in intervals of 100 ns.

If the lifetime is not specified or is unlimited, it sets the 0xFFFFFFFFFFFFFFFF value.

When using a simplified initialization mode, the value is read from the registry, the key is LicenseLifeTimeLimit, the key type is REG_BINARY.

```
unsigned __int32 __stdcall PSA_GetRemainingNumberOfRuns ( unsigned
__int32 *remainingNumberOfRuns );
```

The function defines the number of the allowed program runs left.

If the maximum allowed value is not specified or is unlimited, it sets the 0xFFFFFFFFFFFFFFFF value.

When using the simplified initialization mode, the value is read from the registry, the key is RemainingNumberOfRuns, the key type is REG_DWORD.

```
unsigned __int32 __stdcall PSA_GetLicenseNumberOfRunsLimit
( unsigned __int32 *licenseNumberOfRunsLimit );
```

The function defines the maximum allowed number of program runs.

If it is not specified or is unlimited, the function sets the 0xFFFFFFFFFFFFFFFF value.

When using the simplified initialization mode, the value is read from the registry, the key is LicenseNumberOfRunsLimit, the key type is REG_DWORD.

```
unsigned __int32 __stdcall PSA_GetRemainingExecutionTime ( unsigned
__int64 *remainingExecutionTime);
```

The function defines the total program execution time left, in intervals of 100 ns.

If the total execution time is not specified or is unlimited, it sets the 0xFFFFFFFFFFFFFFFF value.

When using the simplified initialization mode, the value is read from the registry, the key is RemainingExecutionTime, the key type is REG_BINARY.

```
unsigned __int32 __stdcall PSA_GetRemainingExecutionTimeAtStart
( unsigned __int64 *remainingExecutinTimeAtStart );
```

The function returns the total program execution time left, in intervals of 100 ns, calculated at the moment of the program start.

When using the simplified initialization mode, the value is read from the registry, the key is RemainingExecutionTimeAtStart, the key type is REG_BINARY.

```
unsigned __int32 __stdcall PSA_GetLicenseExecutionTimeLimit
(unsigned __int64 *licenseExecutionTimeLimit);
```

The function returns the total program execution time in intervals of 100 ns.

If the total execution time is not specified or is unlimited, it sets the 0xFFFFFFFFFFFFFFFF value.
When using the simplified initialization mode, the value is read from the registry, the key is LicenseExecutionTimeLimit, the key type is REG_BINARY.

```
unsigned __int32 __stdcall PSA_IsTrialMode ( bool *isTrialMode );
```

The function sets the true value for the isTrialMode parameter, if the application is run in trial mode.

When using the simplified initialization mode the value is read from the registry, the key is IsTrialMode, the key type is REG_DWORD.

```
unsigned __int32 __stdcall PSA_GetTimeToLicenseExpiration
( unsigned __int64 *timeToLicenseExpiration);
```

The function defines the time left to license expiration, in 100 ns.

When using the simplified initialization mode, the value is read from the registry, the key is TimeToLicenseExpiration, the key type is REG_BINARY (implemented by 8-byte Binary).

```
unsigned __int32 __stdcall PSA_GetLicenseExpirationDateTime
( unsigned __int64 *LicenseExpirationDateTime);
```

The function defines the date and time of license expiration in FILETIME format (quantity of 100-ns intervals since 1601).

To convert the returned value to other formats, Windows API can be used (i.e. FileTimeToSystemTime, FileTimeToLocalTime).

When using the simplified initialization mode, the value is read from the registry, the key is LicenseExpirationDateTime, the key type is REG_BINARY.

```
unsigned __int32 __stdcall PSA_GetUserDefinedField16Bits ( unsigned
__int16 *userDefinedField );
```

The function returns a field value.

When using the simplified initialization mode the value is read from the registry, the key is UserDefinedField16Bits, the key type is REG_DWORD.

```
unsigned __int32 __stdcall PSA_DisableFeaturesGrantedByLicense
( unsigned __int32 features );
```

The function disables separate bits of the rights mask during the application operation.

The argument affects the enabled rights only. Use -1 argument to disable all the enabled rights. The disabled rights cannot be enabled until the next application launch.

```
unsigned __int32 __stdcall PSA_Uninitialize ();
```

The function de-initializes the protection core before unloading protect.dll or quitting the program.

```
unsigned __int32 __stdcall PSA_GetLicenseInformation ( unsigned
__int32 *version, unsigned __int32 *type, bool *nonCommercial );
```

The function returns special information on the current license.

When using the simplified initialization mode, the value is read from the registry, the keys are LicenseVersion, LicenseType, IsNotCommercialLicense, the key type is REG_DWORD.

```
unsigned __int32 __stdcall PSA_IsLicenseExpired ( bool
*isLicenseExpired, bool updateRunTimeData );
```

The function checks if the license lifetime is expired.

When using the simplified initialization mode, the value is read from the registry, the key is IsLicenseExpired, the key type is REG_DWORD.

3.4.5 Integrating SDK into the application

1. Inlcude `PsaApi.h` (`PscApi.h`) into the source code. `PsConfig.h` is included from `PsaApi.h` (`PscApi.h`).
2. Add `protect.lib` to the list of input files in the linker settings.
3. Copy `protect.dll` and `protect.exe` to the folder the application is to be run from.
4. Add all necessary API function calls to the application being protected.
5. Compile the application.

Notes

1. Files with the `protect` name will actually have the name specified in the project settings.

2. If no protection API functions are used in the application, the linker will perform optimization and will not link `protect.dll` to the application. It is often undesirable because the protection code, that allows modeling its behavior, is executed in `DllMain` of `protect.dll`. This may be solved via using a special method with zero functionality – the void PSA_DummyFunction() function. Its call can be placed anywhere in the application to exclude the linking optimization if no protection API functions are called.
3. For **Visual C++** versions earlier than 2005 a problem of incompatibility between `protect.lib` and the project can occur. In this case it is necessary to create user's own `protect.lib` file. The simplest way to do it is to create your own DLL that has empty functions with the names coinciding with the names of the StarForce API functions. Creating `protect.lib` by creating a `DEF`-file with the subsequent **LIB.EXE** usage is not recommended since it may result in problems with name decoration.
4. To integrate SDK with a **Delphi** application, copy `PsaApi.pas`, `PscApi.pas`, `PsConfig.pas`, `PsConstants.pas` to the folder with the project, and indicate `PsaApi` and `PscApi` in the **`uses`** section in the project. If the protection library name is not `protect.dll`, replace all the '`protect.dll`' string entries in the `PsaApi.pas` and `PscApi.pas` files with the actual protection library name.

Modeling different scenarios of the protected software product behavior by means of SDK

By default, at the startup of the application integrated with `protect.dll`, the full protection initialization (with activation, etc.) is performed in the main function of this library.

It may be inconvenient during the application debugging, so there is a *simplified* `protect.dll` *initialization mode*. To use this mode, it is necessary to create a `Debug` key (the value is QuickInitializationMode, the type is REG_DWORD) in the system registry section, specified in the **General options** section in the **`Key path in the registry`** field. When `QuickInitializationMode=0`, full protection initialization is performed, while for `QuickInitializationMode=1`, the simplified initialization runs.

In the simplified initialization mode, all the protection API functions that return some results use the values from the registry (from the Debug key). Interpretation of the corresponding values is shown in the API function description.

Application protection

The application protection using SDK is performed as usual, but the name of the debug protection library specified during the SDK generation, (`protect.dll` by default) must not be changed. If it is required to change the name, re-generate SDK and use the new protection library name; then modify the protected application in the appropriate way.

Restricting the protected product functionality depending on the license

StarForce protection system allows restricting the functionality of the protected software depending on the license type, on the basis of independently disabled features. This possibility is implemented via an additional field in the Activation Key, which contains the description of functionality provided by the product. This field is called the rights code. Each bit of the rights code corresponds to a set of functions of the protected application, which can be enabled or disabled. The rights code for the current license can be obtained using the

PSA_GetFeaturesGrantedByLicense protection API function. This function is the main method to support the restriction of the functionality of the protected application.

PsApi.h contains independent bit constants, which names correspond to the values used during the protection and the license generation:

```
#define PSA_GrantedFeature0 0x00000001
#define PSA_GrantedFeature1 0x00000002
...
#define PSA_GrantedFeature31 0x80000000
```

Note. Only lower five bits of rights are supported in the protection project, which correspond to the values 0x1 – 0x10.

When necessary, it is possible to assign constants with more reasonable names. Thus, it is possible to define the following in the program:

```
#define PrintFeatureGranted     PSA_ GrantedFeature2
...
unsigned __int32 features;
if( PSA_GetFeaturesGrantedByLicense( &features ) != PSC_STATUS_SUCCESS )
{
      // Process Error
}
if( features & PrintFeatureGranted)
{
    PrintDocument();
}
```

Using API in itself does not provide protection against cracking (unless the API function calls are placed inside the protected functions, which is not always possible to implement). To improve the quality of protection against cracking, it is recommended to make the protected functions depend on the set of rights for the current license (a bit sequence is specified during protection of the function; in this case in order for the protected function to operate normally, it is required that at least one of these rights is set in the current license). If the rights set of the current license does not meet the requirements for the protected function, the function is not executed and it returns 0 to the calling function. It is also recommended to call the protected functions only if their calls are granted by the current license, e.g.:

```
#define PrintFeatureGranted        PSA_ GrantedFeature2
...
unsigned __int32 features;
if( PSA_GetFeaturesGrantedByLicense( &features ) != PSC_STATUS_SUCCESS )
{
      // Processes Error
}
if( features & PrintFeatureGranted )
{
    SFLB_PrintDocument(); // Protected function requiring
PSA_GrantedFeature2
}
```

To ensure the protected functions are called in the appropriate place, it is possible to implement automatic checks for incorrect calls of the protected functions in the debug version of the protection API. To do so, it is necessary to make a check for rights in the beginning of the protected function code:

```
void SFLB_PrintDocument()
{
    PSA_CheckFeaturesGrantedByLicense( PrintRightGranted );
    // Function code
}
```

The `PSA_CheckFeaturesGrantedByLicense` function checks the current rights by comparison of the argument and the return value of `PSA_GetFeaturesGrantedByLicense` and in case of mismatch an error message appears:

```
unsigned __int32 __stdcall PSA_CheckFeaturesGrantedByLicense(
      unsigned __int32 features )
{
      uint32 currentFeatures;
      uint32 result = PSA_GetFeaturesGrantedByLicense( &currentFeatures );
      if( result != PSC_STATUS_SUCCESS )
      {
            return result;
      }
      if( ( features & currentFeatures ) == 0 )
      {
            MessageBox( HWND_DESKTOP, "Error", "Invalid rights", MB_OK );
      }
      return PSC_STATUS_SUCCESS;
}
```

Instead of making a `PSA_CheckFeaturesGrantedByLicense` call, it is also possible to create a user function for this check (which would e.g. write the check result to a file instead of displaying a message). However, it is necessary to note that if `PSA_CheckFeaturesGrantedByLicense` is used during the debug process, its call is automatically deleted from the protected version, while the user function call should be deleted manually.

In order to change the functionality during the application operation, it is possible to use the `PSA_DisableFeaturesGrantedByLicense (int features)`; function.

This function allows disabling separate bits of the rights mask during the application operation. Rights that are disabled cannot be enabled until the next application launch. The function argument affects only the granted features. To disable all granted features, use -1 as the argument value. This function uses `PSA_GetTimeToLicenseExpiration` to find out the license expiration time. If the lifetime has expired, the function disables the rights according to the expired license instead of shutting down the application.

Attention! A standard division by zero processing is not implemented in the SDK (the appropriate exception is not displayed). Division by zero is not usually used in applications. Nevertheless, since there is no such check in the SDK, this check should be performed in a protected function code if division by zero is possible. If it is necessary to display an appropriate exception for some reason, it must be made in the protected function code as well.

3.4.6 SDK example

To illustrate the capabilities of the SDK below is an example of how to use StarForce API to check the integrity of read-only elements of the protected module in memory and read-only files..

```
//====================================================================
//SelfCheckSample.cpp
//--------------------------------------------------------------------
//Description:

// To use this test, compile it with the LIB file from StarForce SDK,
// add a SelfTestSample.txt file to a container, select ‘Use digital
//  signature instead of file’ command, and perform protection
//======================================================================
#include <stdio.h>
#include <windows.h>
#include "PsaApi.h"

// Entry point
void main()
{
	unsigned __int32 status;
	bool checkResultOk;

	// Check memory (1st time)
	printf( "Performing self-check of read-only memory...\n" );
	status = PSA_CheckProtectedModulesReadOnlyMem( &checkResultOk );
	if( status != PSC_STATUS_SUCCESS )
	{
		printf( "PSA_CheckProtectedModulesReadOnlyMem failed with 
error code 0x%08X\n. Terminating application." );
		return;
	}
	if( checkResultOk )
	{
		printf( "Done. Check passed. Read-only memory is valid.\n" );
	}
	else
	{
		printf( "Done. Check failed. Read-only memory is invalid. 
Terminating application.\n" );
		return;
	}
	printf( "\n" );

	// Optional modification of memory
	char option[ 1 ];
	printf( "Do you want to modify read-only memory and check again (y/
n)?" );
	scanf( "%1s", option );
	if( option[ 0 ] == 'y' || option[ 0 ] == 'Y' )
```

```
        {
                printf( "Modifying read-only memory...\n" );
                char *readOnlyData = "Read only data";
                DWORD oldProtect;
                if( !VirtualProtect( readOnlyData, sizeof( char ),
PAGE_EXECUTE_READWRITE, &oldProtect ) )
                {
                        printf( "VirtualProtect failed with error code %d\n",
GetLastError() );
                        return;
                }
                readOnlyData[ 0 ] = 0;
                printf( "Done.\n" );

                // Check memory (2nd time)
                printf( "Performing self-check of read-only memory...\n" );
                status = PSA_CheckProtectedModulesReadOnlyMem
( &checkResultOk );
                if( status != PSC_STATUS_SUCCESS )
                {
                        printf( "PSA_CheckProtectedModulesReadOnlyMem failed
with error code 0x%08X\n. Terminating application." );
                        return;
                }
                if( checkResultOk )
                {
                        printf( "Done. Check passed. Read-only memory is valid.
\n" );
                }
                else
                {
                        printf( "Done. Check failed. Read-only memory is
invalid. Terminating application.\n" );
                        return;
                }
        }
        else
        {
                printf( "Modification skipped.\n" );
        }
        printf( "\n" );

        // Reading data from file
        printf( "Performing self-check of read-only files...\n" );
        FILE *f;
        if( fopen_s( &f, "SelfCheckSample.txt", "rt" ) != 0 )
        {
                printf( "Unable to open file SelfCheckSample.txt\n" );
                return;
        }
        char currentChar;
        printf( "The content of file SelfCheckSample is:\n" );
        while( ( currentChar = fgetc( f ) ) != EOF )
        {
                printf( "%c", currentChar );
        }
```

```
      printf( "\n" );
      int isValid;
      status = PSA_FsVerifyFileSignature( L"SelfCheckSample.txt", &isValid
);
      if( status != PSC_STATUS_SUCCESS )
      {
            printf( "PSA_FsVerifyFileSignature failed with error code 0x%
08X\n. Terminating application." );
            return;
      }
      if( isValid )
      {
            printf( "The file SelfCheckSample.txt was not modified.\n" );
      }
      else
      {
            printf( "The file SelfCheckSample.txt was modified.\n" );
      }
}
```

3.5 Additional protection features

3.5.1 GUI customization

Attention! GUI messages are displayed to an end user in one of the following cases:
1. When installing the driver;
2. When errors occur.

3.5.1.1 Purpose of the customized GUI

Customized GUI allows wide-range changes in the protected application interface and behavior during the check, activation, etc. Below are some typical tasks a user can perform using GUI modification:

- Modification of the window appearance, including modification according to the application design (important for games); using user’s own library to display dialog boxes.
- Addition of auxiliary windows with explanations and warnings.
- Modification of the activation procedure (addition of auxiliary fields, redirection to user’s own website, manual activation disabling, etc.).
- Modification of the procedure to contact customer support, modification of the error report format.

3.5.1.2 Recommendations for the creation of individual GUI projects

To create an individual GUI project:

1. Tick off the ‘Enable GUI customization’ check box in the **SDK** section.
2. Perform the application protection.
3. Generate SDK. The GUI folder in the SDK contains the `UserInterfaceCustomizable.vcproj` file.

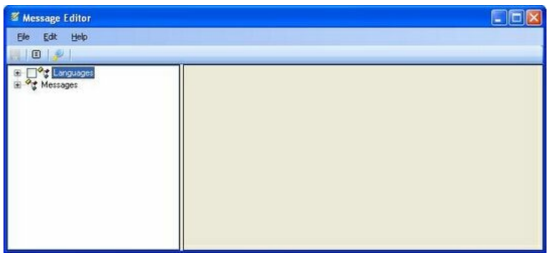
4. Open `UserInterfaceCustomizable.vcproj` using VisualStudio. Reject using SourceControl system when opening the file. Pay attention to the `Config.h` file: it may undergo changes if the project is edited or re-protection is performed. Therefore, it is necessary to re-generate SDK and replace `Config.h` in the project with the new file.
5. Compile the project and save it, which results in generating the `protect.gui` file.
6. Copy `protect.gui` to a folder with `protect.dll` and run the protected application. Make sure it is operable.
7. Make modifications in the project and repeat steps 5 and 6.

3.5.2 Message Editor

Attention! GUI messages are displayed to an end user in one of the following cases:
1. When installing the driver;
2. When errors occur.

3.5.2.1 Message Editor interface

Interface of the Editor is shown in figure below.



Message Editor

Interface of the Editor consists of the following blocks: main menu, toolbar, message/ language tree and the corresponding contextual menu (for languages only), right panel to display information about languages and messages.

3.5.2.2 Options

File menu

Option	Purpose
Save	Save changes
Exit	Close Message Editor

Message Editor. File menu

Edit menu

Option	Purpose
Find	Search among messages
Add language	Add a language

Message Editor. Edit menu

Help menu

Option	Purpose
About Message Editor	Display information about the program

Message Editor. Help menu

3.5.2.3 Toolbar buttons

Button	Purpose
💾	Save changes
▤	Add a language
🔍	Search among messages

Toolbar buttons

3.5.2.4 Using hot keys

Key combination	Purpose
Ctrl+S	Save changes
Ctrl+F	Search among messages
Ctrl+Z	Undo; for entry fields only

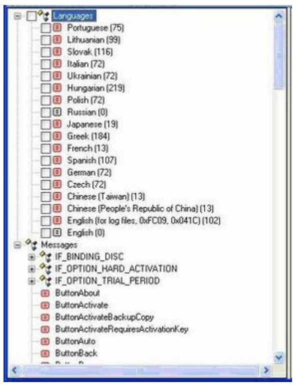
Message Editor. Hot keys

3.5.2.5 Message and language tree

Structure

There are five types of nodes in the message and language tree (see figure):

1. **Languages** – contains a list of languages;
2. **Language** – contains a language;
3. **Messages** – defines message and message group structure;
4. **Message** – contains a message;
5. Message groups.



```
Languages
    Portuguese (75)
    Lithuanian (99)
    Slovak (116)
    Italian (72)
    Ukrainian (72)
    Hungarian (219)
    Polish (72)
    Russian (0)
    Japanese (19)
    Greek (184)
    French (13)
    Spanish (107)
    German (72)
    Czech (72)
    Chinese (Taiwan) (13)
    Chinese (People's Republic of China) (13)
    English (for log files, 0xFC09, 0x041C) (102)
    English (0)
Messages
    IF_BINDING_DISC
    IF_OPTION_HARD_ACTIVATION
    IF_OPTION_TRIAL_PERIOD
    ButtonAbout
    ButtonActivate
    ButtonActivateBackupCopy
    ButtonActivateRequiresActivationKey
    ButtonAuto
    ButtonBack
```

Message and language tree

Highlighting

The tree icons are highlighted as follows:

1. A language name is followed by a number in brackets. The number specifies the amount of the messages marked as '**Todo'** for the selected language (i.e. to be translated or modified). If the number is 0, the icon next to the name is grey, if there are messages that are not translated, the icon is red, which indicates that the work is not finished.
2. The same principle applies to messages: if a message is translated into all languages, its icon is grey, while it is red otherwise.

3.5.2.6 Editor functions

Saving changes

To save a file, **Save** option should be selected (or button should be pressed). After closing the Message Editor it is necessary to save the project file. Thus, all the changes in the messages are saved to the project file.

Loading previously saved messages

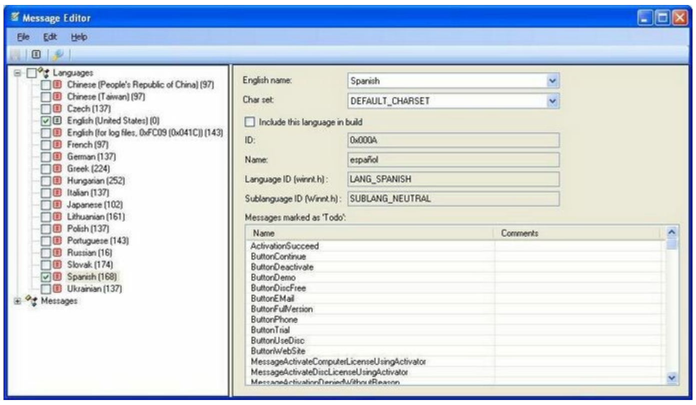
Message loading is performed automatically from the project file after the Editor is run.

Program exit

If one of the fields contains an error, a message appears when exiting the program, offering to exit without saving or resume editing, since saving documents with errors is not permitted.

Adding, editing, deleting languages

To add a new language, press or select **Add language** in contextual menu or in the **Edit** menu. A new language appears in the tree, while fields for entering information about the language appear on the right-hand side of the window (see below). The new language is added to the end of the language tree.



Languages
- [ ] Chinese (People's Republic of China) (97)
- [ ] Chinese (Taiwan) (97)
- [ ] Czech (137)
- [✓] English (United States) (0)
- [ ] English (for log files, 0xFC09 (0x041C)) (143)
- [ ] French (97)
- [ ] German (137)
- [ ] Greek (224)
- [ ] Hungarian (252)
- [ ] Italian (137)
- [ ] Japanese (102)
- [ ] Lithuanian (161)
- [ ] Polish (137)
- [ ] Portuguese (143)
- [ ] Russian (16)
- [ ] Slovak (174)
- [✓] Spanish (168)
- [ ] Ukrainian (137)

Messages

English name: Spanish
Char set: DEFAULT_CHARSET
[ ] Include this language in build
ID: 0x000A
Name: español
Language ID (winnt.h) : LANG_SPANISH
Sublanguage ID (Winnt.h) : SUBLANG_NEUTRAL

Messages marked as 'Todo':

Name	Comments
ActivationSucceed	
ButtonContinue	
ButtonDeactivate	
ButtonDemo	
ButtonDiscFree	
ButtonEMail	
ButtonFullVersion	
ButtonPhone	
ButtonTrial	
ButtonUseDisc	
ButtonWebSite	
MessageActivateComputerLicenseUsingActivator	
MessageActivateDiscLicenseUsingActivator	
MessageActivationDeniedWithoutReason	

Language parameters

When adding a language, the first unused language from the list is added automatically. The required language can then be selected from the **`English name`** drop-down list. Duplication is not permitted when changing a language.

It is recommended to accept the **DEFAULT_CHARSET** default value in the **`Char set`** field.

To include the selected language into a MSG file with the messages, tick off the **`Include this language in build`** checkbox.

Supplemental information on the selected language is displayed in the **`ID`**, **`Name`**, **`Language ID`** and **`Sublanguage ID`** fields.

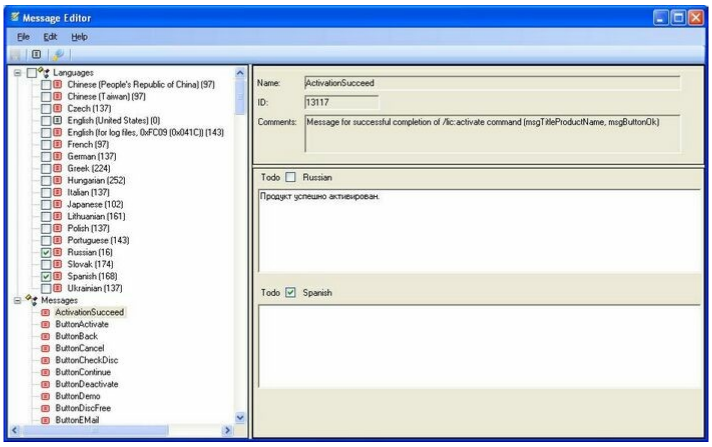
The names of undone messages for the selected language and comments to the messages are listed in the table named **`Messages marked as 'Todo'`**. To start editing a message from this list, it is sufficient to double-click the message.

Editing messages

To edit a message text:

1. Select it in the tree in the **Messages** node;
2. Select the required languages by using the respective check boxes in the tree. After that the fields for editing a message text in the selected languages appears in the message work field (see figure).

After the text is edited, deselect the **`Todo`** checkbox. Deselected **`Todo`** checkbox means that when a language is selected in the protection window, the message is displayed in this language, not in the default language (English).



- Languages
  - ☐ Chinese (People's Republic of China) (97)
  - ☐ Chinese (Taiwan) (97)
  - ☐ Czech (137)
  - ☐ English (United States) (0)
  - ☐ English (for log files, 0xFC09 (0x041C)) (143)
  - ☐ French (97)
  - ☐ German (137)
  - ☐ Greek (224)
  - ☐ Hungarian (252)
  - ☐ Italian (137)
  - ☐ Japanese (102)
  - ☐ Lithuanian (161)
  - ☐ Polish (137)
  - ☐ Portuguese (143)
  - ☑ Russian (16)
  - ☐ Slovak (174)
  - ☑ Spanish (168)
  - ☐ Ukrainian (137)
- Messages
  - ActivationSucceed
  - ButtonActivate
  - ButtonBack
  - ButtonCancel
  - ButtonCheckDisc
  - ButtonContinue
  - ButtonDeactivate
  - ButtonDemo
  - ButtonDiscFree
  - ButtonEMail

Name: ActivationSucceed
ID: 13117
Comments: Message for successful completion of /lic:activate command (msgTitleProductName, msgButtonOk)

Todo ☐ Russian
Продукт успешно активирован.

Todo ☑ Spanish

Editing messages

Search

The search window appears when pressing or when selecting **Find** option in **Edit** menu. The search window also appears when pressing **Ctrl+F**.

The search is only performed among messages, namely:

- among message names,
- among message IDs,
- among comments to messages,
- among message texts.

The search results are displayed as follows.

1. Only first entry in a string is highlighted. The search for other entries in the string is not performed.
2. If a language which an entry is found for is hidden, then this language is checked automatically and is displayed. However, the languages that were checked previously are not hidden even if the text is not found for them.
3. The entries that were found in other strings are also highlighted at the same time.

The sequence of operations for the search function:

1. When opening the search window for the first time, the **Find First** button is disabled. It is enabled after a text is entered in the **`Find what`** field.
2. If a text is found, the **Find First** inscription changes to **Find Next**. By pressing **Find Next** the search begins from the next message in the tree (while taking nesting into account).
3. If a text is not found, the appropriate message appears. The search is not performed from the beginning.
4. When selecting a node in the tree, the search is performed starting from the selected node (**Find First**) or from the next one (**Find Next**).
5. If a text is found, but the text in the **`Find what`** field is changed, the **Find Next** inscription changes to **Find First**, and the search begins from the selected tree node.

Attention! The search window does not save the query after the closure.

6. Only one search window can be opened. When the search window is opened, the **Find** toolbar button becomes disabled, and **Ctrl+F** hot key does not work either. After the search window is closed, the **Find** button becomes enabled.

4 Preparations for the release and product release

4.1 Redefining the protection system settings

Certain options defined in the project file (i.e. at the moment the protection is applied to the product) can be redefined by writing the corresponding values to the system registry in an end user's computer. In this case, the protection system reads the values from the registry and only if there are no values in the registry, it uses the values defined during the protection. Writing values to the system registry is usually made during the product installation on an end user's computer via the corresponding operations included into the installation scripts.

Each protected application has its own section in the system registry. The full name of the section is defined in the **General Options** section of Protection Studio 'General options' section).

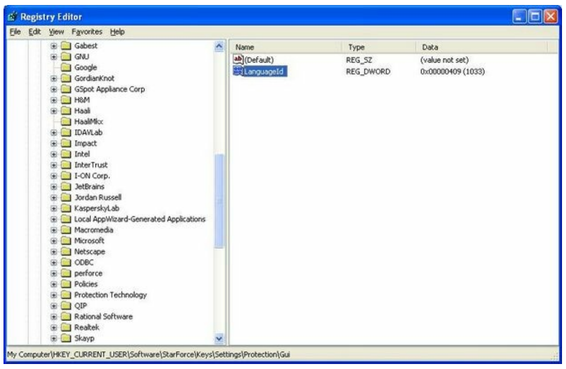
The default name of the registry section is `HKEY_CURRENT_USER/SOFTWARE/<company name>/<project name>/Keys` (a section name can only contain the following characters: Latin letters, figures, \ ( ) { } [ ] . : underline, space, hyphen). The `Settings` subsection can be created in this section (for example, during the application installation). This subsection is used to redefine the following protection parameters:

1. Parameters with text data are redefined with the values of the REG_SZ type according to the table below:

Protection Studio section	Parameter name in Protection Studio	Key in the registry	Value name
Protection GUI options	Protection GUI messages' field: Support/Web site	`Protection\Gui\Support`	`Page`
Protection GUI options	'Protection GUI messages' field: Support/E-mail	`Protection\Gui\Support`	`Email`
Protection GUI options	'Protection GUI messages' field: Support/Phone	`Protection\Gui\Support`	`Phone`
Project	Company name	`Protection\Gui\About`	`Company`
Project	Company name	`Protection\Gui\About`	`Product`

Text parameters

2. To redefine the language of the interface, the Protection\Gui registry key with the LanguageId value of the REG_DWORD type is used.
   Available language identifiers correspond to the values of the Microsoft Windows LANGDID type. Available interface languages are indicated in the '**Protection GUI option**' section as the values of the **GUI options|Language** option (see figure). If a specified language is not supported, the language is selected according to the settings in the **Regional and language options** section of the OS.
   The exact ID of the language can be found in the Message Editor (see figure, **`ID`** field).
   An example of a registry record for the English language of the interface is shown in figure below (a registry editor can be invoked from the **Start|Run|regedit** menu).



Name	Type	Data
(Default)	REG_SZ	(value not set)
LanguageId	REG_DWORD	0x00000409 (1033)

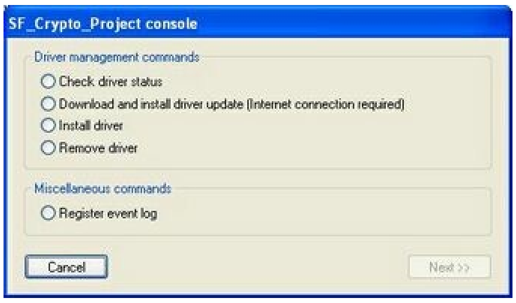
My Computer\HKEY_CURRENT_USER\Software\StarForce\Keys\Settings\Protection\Gui

A registry record for the key that defines the language of the interface

4.2 Protected product installation option

An executable file of the protection interface, `protect.exe`, is generated during the operation of the **Protect files** service (see **Protection of files**) (or a file with the name, which the '`protect`' name was replaced with in the project file settings). It is possible to start working with this file in one of the two ways:

1. Running the protection file directly.
   After that the "Protection System" window is displayed with the list of operations this file can perform. The most complete version is shown in figure below.



SF_Crypto_Project console

Driver management commands

- ○ Check driver status
- ○ Download and install driver update (Internet connection required)
- ○ Install driver
- ○ Remove driver

Miscellaneous commands

- ○ Register event log

Dialog box of the protect.exe protection file

If the driver is not used in the project, the Driver management commands group is not displayed.

2. Running the protection file from the command line with one of the keys.

   This makes it useful when installing/de-installing the protected product on an end user's computer.

Below is a list of keys that can be used to run `protect.exe` in the installation/de-installation scripts, and the correspondence to the commands in the dialog box.

Key	Command	Purpose
`/drv:check [/nogui]`	Check driver status	Checks if the protection driver is installed. The return code is 0 if the driver is not installed, while it is non-zero otherwise.
-	Download and install driver update	Downloads the update from the website and updates the driver.
`/drv:install [/nogui] [/forcereboot]`	Install driver	Installs the protection driver.
`/drv:uninstall [/nogui]`	Remove driver if it is not used by other installed applications	De-installs the protection driver if it is not used by other installed protected applications. If the driver is used by other protected applications, it remains in the system.
`/drv:remove [/nogui]`	Remove driver completely	De-installs the protection driver irrespectively of the presence of the applications using the driver, in the system..
`/? или /help`	-	Displays Help for the keys in use.
`/eventlog:register`	Register event log	Creates a registry section for the Event Log of the protected application-service.

Commands and keys to run the protect.exe protection file

Besides the keys for the main operations, two additional keys can be also used in the command line:

Key	Purpose
`/nogui`	If this key is not specified, the messages of the executable file of the protection are displayed in the standard protection interface. Otherwise, all the information is only available in the return code.
`/forcereboot`	Enables force reboot of the computer after the installation.

Additional keys to run protect.exe

The values of the return codes and the error codes are listed in the table below:

Error type	Return code	Description
`DRVMAN_SUCCESS`	0	Driver installation/de-installation is performed successfully.
`DRVMAN_ERROR`	1	Unexpected error has occurred during the driver installation/de-installation – it is necessary to invoke GetLastError() to obtain the detailed error information.
`DRVMAN_NEED_ADMIN`	3	Error. Driver installation/de-installation requires administrator rights for this computer.
`DRVMAN_USAGE_CONFLICT`	4	Error. For the installation/de-installation, it is necessary to close all running applications and invoke the function once more.
`DRVMAN_DEBUGGER_DETECTED`	5	Error. A debugger is detected, it is impossible to continue installation.
`DRVMAN_INCOMPATIBLE_OS`	6	Error. The protection system is incompatible with the operating system.
`DRVMAN_SAFE_MODE`	7	Error. Windows is running in the Safe Mode. It is impossible to continue installation.
`DRVMAN_COMPATIBILITY_MODE`	8	Error. An attempt to run the application in the Compatibility Mode is made.

Error types and return codes

5 Product maintenance

5.1 Protection driver

For each product protected with driver, a unique driver is created during the protection. The digital signature of the driver is performed by the protection builder; however, a user (a publisher) can apply his own digital signature to it.

Whether to use the protection driver or not is decided by the user on the stage of the protection project co-ordination with StarForce customer support; and then it cannot be changed without contacting StarForce customer support.

The availability of the protection driver is enabled during the creation of the workspace – protection project, and is displayed in the `Driver` field in the **General options** section (see **'General options' section**). This field can have the following values:

Value of the `Driver` field	Available protection options
`Enabled`	Anti-emulation is enabled. It is also possible to apply data file protection (creation of the containers).
`Disabled`	Data file protection is supported in this project.

'Driver' field

5.1.1 Installing driver on an end user's computer

If the driver installation is not supported by the publisher in the installation script, the user is suggested to confirm driver installation when running the application for the first time. Displaying this suggestion is connected with two GUI options: **`Driver installation question`** and **`Display driver FAQ`** (see **'Protection GUI options' section**).

To install driver during the installation of the protected software product, the executable file of the protection interface should be used (see **Protected product installation option**). When running this file as a separate process with different keys (e.g. from the installation script), it is possible to install and remove the protection driver.

For the convenience of the end users, and to increase the compatibility of the protected products, it is recommended to provide the end users with the following information (or links to it in the Internet):

http://www.star-force.com/faq

This webpage contains detailed information on the StarForce protection driver, as well as the StarForce-developed tools for manual update and removal of the driver and information on how to use them.

5.2 Update of the protected product. Protection of the updated versions

Any files that should replace files of previous product version or should be added into the previous version can be called the *updates*.

To protect the updates, perform the following operations:

1. Load the protection project file of the product version to be updated.
2. In the **Files** section: add new executable files, replace old file versions with the new ones and change the paths of the unchanged files if necessary. If the updated executable files have the same paths, these updated files are used automatically during the protection.
3. When necessary, set the protection of the module functions and specify other protection settings.
4. Create a container (or containers) with a name different from the already used container names for this product, and add only new and changed data files into it (them). Tick off the names of old containers in the 'Container manager' window (see figure).
5. Perform file protection (run the **Protect files** service).

Attention! The protection settings ***must not*** be modified for the containers that are already included in the loaded project (i.e. for containers protected in the previous version.

After the service operation completes, the following (updated) files are obtained:

- protection modules (`PROTECT.DLL`, `PROTECT.EXE` by default);
- `PROTECT.X86/X64` files;
- protected executable files of the software product, where these files use the updated data;
- old and new containers.

In order for the application installed on an end user's computer to use the updated data, it is necessary to place the new containers into the relevant folder of the application (the old containers should retain their location; otherwise, the application will not run). In case if there are several containers for the protected software product, the later created container will be of higher priority. If there is the same data file in several containers, it will be read only from the container with the higher priority.

A new data container can be easily distributed, e.g. via the website of the software product developer (in this case there is no need to distribute old containers).

6 Troubleshooting

6.1 Troubleshooting during the protection implementation

All operations are displayed in the **Log** tab.

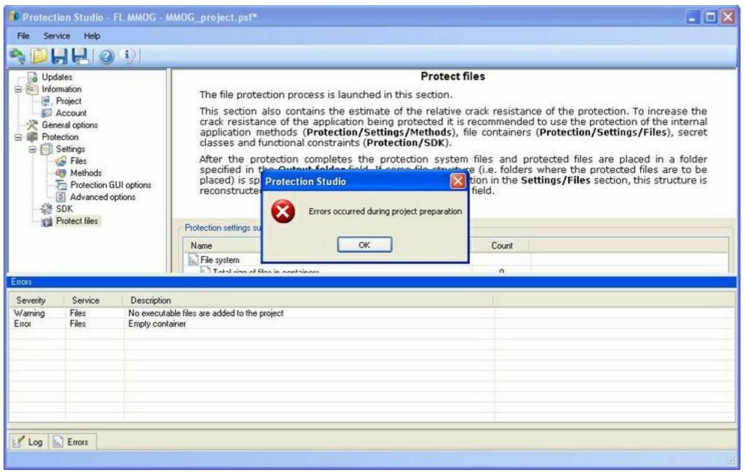
If wrong actions are performed during the protection parameters setting, Protection Studio gives an alert. For example, if the name of a folder to take a file for the protection from is invalid, the names of the file and the folder are highlighted red in the work field of the **Files** section, which means an error. However, the validation is not performed for all the data during the input.

When a service is run, the mandatory check of the parameters specified for this service is performed at first. If everything is correct, the service runs, and a tab with the service name appears in the lower part of Protection Studio window. The tab contains information about the progress of the service operation. Besides, the progress of the service operation is displayed in the **Log** tab.

Errors that occur during a service operation are displayed and described either in the service tab or in the **Errors** tab depending on the error type. The **Errors** tab contains only the errors of parameters setting in Protection Studio prior to the service launch. The errors are divided into *Warnings* and the *Errors* themselves depending on how serious an error is. *Warnings* do not prevent service launch and operation, while it is impossible to run a service with an *Error*.

If the errors in settings are detected when trying to run a service, the service does not begin operation, and a message box is displayed with the following message: "Errors occurred during project preparation". The **Errors** tab displays information on how serious an error is and the name of a section where the error occurred, as well as the error description. The tab for the service to be run is not displayed in this case.

For example, when trying to protect a project without an executable file and with an empty container, the protection process does not start. Therefore, the **Protect** tab is not displayed (see **Protection of files**) as shown in figure below.



Protect files

The file protection process is launched in this section.

This section also contains the estimate of the relative crack resistance of the protection. To increase the crack resistance of the application being protected it is recommended to use the protection of the internal application methods (**Protection/Settings/Methods**), file containers (**Protection/Settings/Files**), secret classes and functional constraints (**Protection/SDK**).

After the protection completes the protection system files and protected files are placed in a folder specified in t… e (i.e. folders where the protected files are to be placed) is sp… tion in the **Settings/Files** section, this structure is reconstructe… field.

Protection Studio

Errors occurred during project preparation

Errors

Severity	Service	Description
Warning	Files	No executable files are added to the project
Error	Files	Empty container

Error during the protection implementation

If a service runs successfully, one session of the service operation is displayed in the corresponding pop-up window. If it is necessary to run the service once more, its tab should be closed. Information about all the operations performed during the complete user session in Protection Studio is saved to the **Log** window.

If problems with the protection implementation occur, additional debugging information may be useful. The generation of this information is enabled in the ‘Options’ window (see **Menu commands**).
Table shows the errors that occur most frequently during the protection implementation and the connection to server:

Error message	Possible reason	Recommendations for troubleshooting
Connection to the remote host has been closed according to mutual agreement.	Network or server failure.	Try to connect to server again; otherwise, contact customer support.
Failed to compose output module <module name>. Try to change extra data handling options.	The size of the executable module part has increased during the protection. As a result, the overlay was not placed to the original address as specified by the ‘Preserve tail data at the same position’ value of the **Overlay position** option.	Check that the file is neither packed nor protected by other protection systems, or contact customer support to perform such check and change the project settings for the file.
File is already protected by Protection System.	An already protected module is selected as a file for protection.	Update the path to this executable file, so that it points to the original (unprotected) version.
Function: <name> can not be converted to pcode. Possible reasons: function contains shared blocks, does not have convertible instructions.	The function cannot be protected; it is possible that it contains shared blocks, or it does not have instructions that can be protected.	Disable protection for this function and select another function; otherwise, use a PDB file.
No active protection projects found to connect to. Contact sales department to extend/ unblock the projects needed and try again.	All the protection projects connected to this account are either expired or blocked.	Contact sales department to extend/unblock the required projects, and try to connect to protection server again.

Error message	Possible reason	Recommendations for troubleshooting
Project parameters cannot be retrieved from the server. Corresponding schema is not found. Contact customer support (support@star-force.com).	Project version or scheme has been changed.	Contact customer support to update the project version or scheme.
The server is busy now. The number of connected users is exceeded. Try again later or connect to another protection server.	Server is busy because of the maximum number of the connected users.	Try to connect to the server later or connect to another protection server.
This operation cannot be performed since the corresponding project's quota has been exceeded: {0}. Contact customer support (support@star-force.com) to increase the quota.	The quota for this operation (the number of times the operation can be performed) has been exceeded for this project.	Send a request to StarForce sales manager to increase the quota for this operation for the project.
This workspace has expired. To extend it, contact sales department.	The lifetime of the user workspace on the StarForce protection server has been expired.	Send a request to the sales department to extend the lifetime of the workspace.

Errors during the protection implementation

Protection system behavior features

- When trying to move a file by the protected program from a disk to where a file in a container is placed, the protection system displays the ERROR_ACCESS_DENIED (5) error message instead of ERROR_ALREADY_EXISTS (183).
- When trying to create a new directory, using the protected program, with the name of an already existing directory from a container, the protection system displays the ERROR_ACCESS_DENIED (5) error message instead of ERROR_ALREADY_EXISTS (183).
- When trying to create a new subdirectory inside a directory already existing in a container, the ERROR_PATH_NOT_FOUND(3) error message is displayed instead of ERROR_DISK_FULL (112).

If there are any problems when debugging the errors associated with the protection implementation, contact StarForce customer support (see Customer support).

6.2 Troubleshooting during the protected application startup

The table enumerates the errors that occur most frequently during the protected application startup, as well as recommendations for troubleshooting and message IDs in Message Editor.

ID	Error message	Possible reason	Recommendations for troubleshooting
MessageBadProcess	**An attempt to run the application from illegal**	An attempt to run the application from an illegal	Close the software and restart the application.

ID	Error message	Possible reason	Recommendations for troubleshooting
Owner	**software is detected. Close the software and restart the application. If the problem recurs, press "Error report" and send the report to the product Customer Support.**	software.	
MessageCanNotRunUnderDebugger	**This application cannot be run under debugger. Deactivate all active debuggers and run the application again.**	A debugger(s) is running.	Deactivate all running debuggers and run the application.
MessageDataImport	**Unable to run the application. The protected module %1!s! contains protected data imports. Protection implementation error.**	Data imports are detected during loading.	Send the error report to the product Customer support.
MessageDriverOpenFailed	**Internal error when accessing the driver. Error code: %1!d!.**	Incorrect operation with the protection driver.	Send the error report to the Customer support.
MessageDriversAreNotInstalled	**Driver is not installed on your computer or is installed incompletely.**	The protection driver is not installed on a user's computer or is installed incompletely.	Install the protection driver completely.
MessageDriversUpdateNetworkError	**Unable to connect to the update server. Check your Internet connection and try again. Error code: %1!d!. To obtain an error report and send it to product technical support, press the corresponding button.**	Unable to connect to server to update the driver.	Check Internet connection and try again.
MessageExpiredWithNoBinding	**The application lifetime is expired.**	The application lifetime has expired.	Contact customer support to purchase a new product.
MessageFileSystemInitFailed	**Error when accessing the application data. Reinstall the application and try again.**	System failure.	Reinstall the application.
MessageInternalError	**Internal error (error code: 0x%1!X!). Close all applications and try again. If this does not help, press "Error report" and send the report to product customer support.**	System failure.	Send the error report to the Customer support.
MessageInvalidDll	**Unable to load the dynamic link library file: %1!s!.**	Some dll files cannot be run on the user's computer.	Send the error report to the Customer support
MessageInvalidTimeSettings	**Unable to run the application due to incorrect time settings. Check the time settings and restart the application. If this**	Incorrect time settings in the system.	Check the system time settings and restart the application.

ID	Error message	Possible reason	Recommendations for troubleshooting
	problem persists, press "Error report" and send the report to product technical support.		
MessageModuleInfected	**The application module %1!s! was modified. It may be damaged or infected by a virus. Scan your computer for viruses and reinstall the application.**	One of the protected modules or protection library are damaged or infected by a virus.	Scan the computer for viruses and reinstall the application.
MessageMultipleInstancesConflict	**The application is already running. Unable to run the second instance of the application.**	An attempt to run the second instance of the application.	Use the running application or close and restart it.
MessageNeedAdministrator	**To install or remove the driver, you must have Administrator privileges on your computer. To obtain an error report and send it to product technical support, press the corresponding button.**	Administrator privileges are required to install/remove driver.	Make sure you have administrator privileges and try again.
MessageNeedCloseAllPrograms	**Another program is using a different version of the driver. Close all windows and restart the application.**	The protection driver is incompatible with the drivers used by another program.	Shut down all the unnecessary programs and restart the application.
MessagePcEmulatorDetected	**An attempt to use a virtual PC is detected. Please close all PC emulators and try again.**	An emulator is running on the user’s computer.	Close all the emulating programs and restart the application.
MessageRegistryAccessError	**Unable to access or modify the system registry. Make sure you have privileges to write to the system registry. If the error recurs, press "Error report" and send the report to product technical support.**	No permissions to access the system registry.	Check for the permissions to modify the system registry and try again.
MessageRemoteSessionDetected	**The application cannot be started remotely. Run the application from your local computer.**	The protection version does not support the application startup on a remote computer.	Run the application from the local computer.
MessageSafeMode	**Windows is running in the Safe Mode. The application cannot be run. Reboot the computer in the normal mode. If the error recurs, press "Error report" and send the report to product technical support.**	Windows is running in Safe Mode.	Reboot the computer in a standard mode and try again.
MessageShellExecuteDoesNotWork	**Quick View Plus may be installed on your computer, which results in incorrect operation of the protection**	Quick View Plus is installed on the computer.	Disable the following option: View → Configure Quick View Plus → Applications → Windows Explorer → View

ID	Error message	Possible reason	Recommendations for troubleshooting
	system. To solve the problem, run Quick View Plus and disable the following option: View → Configure Quick View Plus → Applications → Windows Explorer → View Unregistered File Types.		Unregistered File Types and try again.
MessageUnsupportedOperatingSystem	**Your operating system does not meet the system requirements of the product.**	The operating system on a user's computer does not meet the system requirements of the product.	Reinstall the system. Contact customer support to determine the version.
MessageUnsupportedProcessor	**To run the application, Pentium (tm) or any other compatible processor is required. To obtain an error report and send it to product technical support, press the corresponding button.**	The processor in your computer does not support the application.	Use a computer with another processor.

Errors during the protected application startup

When specifying the protection settings, it is possible to change the number of messages and the amount of information in them. A number of options in the **Protection GUI options** section is used to do so, namely: **`Driver installation question`**, **`Driver installation FAQ`**, **`Show Help`**, **`Show progress bar`** (see figure).

Besides, there is a possibility to enable logging the information to the event log, using the `protect.exe` protection file. To do so, it is necessary to select **`Register event log`** in the file dialog box (see figure).

An end user is suggested to refer all his questions to the product customer support. The contact details should be indicated in the corresponding entry field in the **Protection GUI options** section (see figure).

6.3 Recommendations on the protection library file allocation

If problems occur when running the protected application, we recommend to change the location of `PROTECT.DLL` и `PROTECT.EXE`. Possible ways to allocate these files depending on the application structure are described below.

PROTECT.DLL and PROTECT.EXE allocation

If the protected executable files are not in a single folder in the installed application the question is how to search for the `PROTECT.DLL` protection library and the `PROTECT.EXE` GUI module. Since each protected module imports methods from `PROTECT.DLL`, this library should be loaded to memory when the protected module is loaded. `PROTECT.EXE` should be placed so that `PROTECT.DLL` can find it.

Example. Two files, `protected1.exe` and `protected2.dll`, are being protected. They are located as follows.

```
ROOT_FOLDER
```

```
    SUBFOLDER1
       protected1.exe
    SUBFOLDER2
       protected2.dll
```

The question is how to place `PROTECT.DLL` and `PROTECT.EXE` after protection for the application to operate properly?

By default, Protection Studio places one instance of `PROTECT.DLL` to each folder containing at least one protected module, and a pair of `PROTECT.DLL` + `PROTECT.EXE` to Root folder. Because of such allocation, the system can find the required files almost always; however, there is a redundancy. Besides, if the files are allocated this way, problems can occur when running the application in some cases (which cannot be determined during the protection implementation).

As a result, sometimes it is necessary to place these files manually.

Below are the considerations that should be kept in mind when allocating `PROTECT.DLL` and `PROTECT.EXE`, as well as an example when standard file allocation results in inoperative application (a method to solve the problem is described).

The first consideration that should be addressed when allocating `PROTECT.DLL` is the default Windows DLL search algorithm. Let us suppose that the application does not redefine this procedure. Redefining DLL search procedure is applied to support complex plug-ins containing several DLLs. Most applications do not redefine the standard search procedure.

The default Windows DLL search procedure is as follows.

1) a folder that contains an EXE module initiating the process;

2) system directory (usually C:\Windows\System32);

3) Windows directory (usually C:\Windows);

4) current folder;

5) folder, specified in the PATH environment variable.

Note. In Windows version earlier than XP SP2 'current folder' is the second item in the list.

On the basis of item 1), it is correct to place `PROTECT.DLL` along with an EXE file that identifies the process. In the above-mentioned example, let us suppose that `protected1.exe` is the only EXE file in the application and no other files load `PROTECT.EXE`. Then `PROTECT.DLL` and `PROTECT.EXE` should be placed as follows.

```
ROOT_FOLDER
    SUBFOLDER1
       protected1.exe
       protect.dll
       protect.exe
    SUBFOLDER2
       protected2.dll
```

If the application being protected has several EXE files in different folders, `PROTECT.DLL` should be placed to each of these folders irrespectively of whether an EXE file in a folder is protected or not:

```
ROOT_FOLDER
   SUBFOLDER1
      protected1.exe
      protect.dll
      protect.exe
   SUBFOLDER2
      protected2.dll
   SUBFOLDER3
      protected3.exe
      protect.dll
      protect.exe
   SUBFOLDER4
      not_protected4.exe
      protect.dll
      protect.exe
```

Since the folder structure is specified in Protection Studio during protection, `PROTECT.DLL` can always find ROOT_FOLDER. This saves us the trouble of storing several instances of `PROTECT.EXE`: it can simply be moved to ROOT_FOLDER. However, we have to place an instance of `PROTECT.DLL` along with `PROTECT.EXE` for the methods of `PROTECT.EXE` to operate when `PROTECT.EXE` runs without parameters (activation, driver installation, etc.). The folder structure is then as follows.

```
ROOT_FOLDER
   protect.dll
   protect.exe
   SUBFOLDER1
      protected1.exe
      protect.dll
   SUBFOLDER2
      protected2.dll
   SUBFOLDER3
      protected3.exe
      protect.dll
   SUBFOLDER4
      not_protected4.exe
      protect.dll
```

If it is certain that the protected application will always run with the indication of a certain current folder, it is possible to leave only one instance of `PROTECT.DLL` + `PROTECT.EXE` and place it to this folder:

```
ROOT_FOLDER
   CURRENT_FOLDER_FOR_ALL_RUNS_FOR_ALL_EXES
```

```
      protect.dll
      protect.exe
   SUBFOLDER1
      protected1.exe
   SUBFOLDER2
      protected2.dll
   SUBFOLDER3
      protected3.exe
   SUBFOLDER4
      not_protected4.exe
```

If only DLLs are protected in the application, `PROTECT.DLL` should be placed along with an EXE file (unprotected in this case) rather than a DLL file.

```
ROOT_FOLDER
   SUBFOLDER1
      protected1.dll
   SUBFOLDER2
      not_protected2.exe
      protect.dll
      protect.exe
```

Now let us suppose that the standard DLL search procedure is redefined by a developer. Such redefining is performed using the LoadLibraryEx method to load DLLs. (The search procedure for DLLs specified in the imports table cannot be modified; therefore, if all the protected DLLs are only loaded through the imports tables, the standard procedure is always applied.)

The only difference between the redefined procedure and the standard one is that a folder with DLL being loaded by LoadLibraryEx is used instead of the folder with an EXE file that initiates the process (i.e. 1st item from the list above). This modification of the algorithm allows a plug-in consisting of several DLLs to find all these DLLs correctly, irrespectively of the location of the calling EXE file. (If the standard search algorithm is used, it is impossible to find additional plug-in DLLs .)

Therefore, when protecting plug-ins that are loaded by LoadLibraryEx, `PROTECT.DLL` should be placed along with the main plug-in DLL being loaded:

```
ROOT_FOLDER
   PLUGINS
      protected1.dll
      protect.exe
```

However, a number of collisions can occur:

First, a plug-in is not necessarily loaded by LoadLibraryEx (it is possible that a developer that uses the plug-in has not considered a complex plug-in). Then the system may not find `PROTECT.DLL` placed along with the plug-in.

Second, if both the plug-in and the application that uses it are protected, this complicates the matters further. Let us suppose the folder structure is as follows.

```
ROOT_FOLDER
   protected1.exe (loads protected2.dll with the help of LoadLibraryEx)
   PLUGINS
      protected2.dll
```

Then if `PROTECT.DLL` is only placed along with `protected1.exe`, `protected2.dll` cannot find it. If `PROTECT.DLL` is only placed along with `protected2.dll`, `protected1.exe` cannot find it. If `PROTECT.DLL` is in both locations, it is loaded to memory twice (since it is loaded from different folders), which results either in double check or in error: MULTIPLE_INSTANCES_CONFLICT.

There is no all-purpose recommendation for such complicated cases. The simplest method is the requirement to always run the application from the same current folder (and place `PROTECT.DLL` in this very folder). Modification of the PATH environment variable can be considered as well.

Auxiliary files such as `PROTECT.X86`, `PROTECT.X64`, `PROTECT.MSG`, and `PROTECT.GUI` should be placed along with `PROTECT.EXE`. One instance of each file in the root project folder usually suffice.

7 Glossary

Term	Description
Account	A set of parameters, which uniquely identifies a certain user in StarForce system: username (login) and password. The account gives its owner the right to perform certain actions to protect products within the limits of workspaces assigned to it, and according to the account role set in the project. The account also contains some additional information about its owner. For example, the email specified in the account is used to dispatch messages announcing some events concerning this account.
Application to be protected	Part of the product for protection including files, which are to be protected directly.
Binding	A protection method based on the integration of a direct or indirect link with a physical license media into an application being protected.
Cracking	A method intended for unbinding the protected application from the physical license media and from the protection core by means of extraction and neutralization of the protection block from the code of the protected modules by disassembling and step-by-step debugging. Besides, during cracking the original parts of protected module code are restored in order to achieve the full application functioning without the object of binding.
End-user	A user of products protected with the protection system.
FL MMOG	The FL MMOG product provides protection of game servers against reverse engineering, modification and running servers on unauthorized sites; protection of the game code against reverse engineering and cracking; protection against running and executing cheat programs and bots; protection of traffic between an MMOG server and a client.

Term	Description
Graphical user interface	A part of the protection system needed to provide a dialogue with a user of the protected application. Protection GUI is represented by a set of typical windows displaying information about errors etc.
Module	Relatively independent part of some system performing its own definite function.
Period of validity of StarForce protection project	A period calculated from the moment of opening the corresponding StarForce project, during which the protection of the new versions of the corresponding product is done without extra charge.
Product to be protected	A product to be protected with StarForce products against unauthorized use.
Project container	A workspace giving a possibility to create StarForce protection projects for users assigned to it.
Protection	A set of actions aimed at counteracting the illegal use of products to be protected.
Protection core	A protection system component implementing the means of protection against cracking, reverse engineering and emulation during the operation of the protected application.
Protection driver	An executable module which is a part of the system of protection against industrial piracy. Protection driver uses low-level Windows tools to prevent the attempts of cracking and emulation.
Protection level	Complex measure determining the degree of resistance of the protected application to cracking, emulation and reverse engineering.
Protection method	A set of operations performed to modify the protected application with the aim of providing its protection against unauthorized use.
Protection of imported functions	Extraction of references to the imported functions from the module to be protected and their placement into the protection core. There is an automatic protection of imports (that is included into the encryption of modules), as well as the additional protection of imports, which allows increasing the protection level insignificantly in cases when there is no access to the protected application code and when protection of internal functions is impossible.
Protection of internal functions	The process during which the code of the function marked for protection is partially extracted from the module and is then placed into the protection core in a modified form. Protection of internal functions is the most effective way to increase the reliability of the application protection.
Protection of modules	Modification of the application modules aimed to provide protection against cracking and reverse engineering. The protection of modules can include one or several means of counteracting cracking and reverse engineering: 1. Protection of internal functions 2. Protection of imported functions 3. Encryption of modules 4. Protection of the application data
Protection of the application data	The process of placing the data files in a special file container in an encrypted form. The protection of data files is possible only when protecting the executable module that accesses these files.

Term	Description
Protection project file	A file containing the protection settings of a specific version of an application for a certain protection project.
Protection Studio (PS)	A software complex designed to implement protection on the applications.
Quota	A limit on the number of certain actions for the protection of a product (file protection, etc.) Quotas are valid within a workspace and are available only for users of this space; but they can be distributed among other users by creating subspaces.
Reverse engineering	Analysis of the protected application by means of disassembling or step-by-step tracing for the purpose of revealing the algorithms of operation of its separate functions.
Role	A set of static rights defining the range of a user's possible actions in the system relatively to a certain workspace. One user may be appointed several unique roles in the same workspace; in this case the rights authorized by each of the roles are summed up.
SDK **Software Development Kit**	Protection SDK is a set of tools for complex integration of the protection into a product to be protected.
StarForce protection project	A workspace defining a set of protection settings of a certain product to be protected.
StarForce protection system **Briefly: Protection system**	A combination of technical means implementing the protection of software products. Protection implementation is performed by means of the Protection Studio software system.
User	Juridical or natural person (a client, a customer) possessing the right to use the protection system.
Version of StarForce	A version of the product represented as a set of certain functional capabilities. The protection version is determined by the version of the Protection Studio and the version of server services used during the application protection.
Workspace	A combination of informational objects available for the user of the StarForce protection system. The types of workspaces are a container of projects, a project.

Glossary

8 Customer support

In case if there are questions and/or troubles during work with Protection Studio and with the protection system in the whole, contact StarForce customer support:

- Email: support@star-force.com
- Phone: +7 (495) 967 14 53

Index