Applied Computer Science, vol. 17, no. 2, pp. 5–13
doi: 10.23743/acs-2021-09

Submitted: 2021-05-13 | Revised: 2021-06-14 | Accepted: 2021-06-20

automation controller, COM object, authorization

Denis RATOV [0000-0003-4326-3030]*

INTEGRATION WITH THE SOFTWARE INTERFACE OF THE COM SERVER FOR AUTHORIZED USER

Abstract

The article is devoted to the development of a software controller for automation of access to tools and object model of the multifunctional graphic editor Adobe Photoshop. The work of the graphic editor is initiated in the form of a COM object, which contains methods available to the software controller through the COM interface, which allows the software to use the functionality of the editor. To restrict unauthorized access, a software authorization control protocol is proposed, which is based on the use of binding to the computer hardware and encryption by a 128-bit MD5 public key hashing algorithm.

1. INTRODUCTION

Comprehensive implementation and development of information technology (Ratov, 2020a; 2020b) has led to the need to automate processes in many software packages that are widely used. In addition, to maintain the confidentiality of data from unauthorized access for such a system requires the use of network software.

The purpose of the work is the creation of a protected from unauthorized access software controller to automate the process of document formation, which interacts with the COM interface of the multifunctional graphic editor Adobe Photoshop.

An example of a server in the form of a file that is executed and which implements the ability to create COM objects at the request of other programs are Microsoft Office applications (Artemov, 2007). COM-server, in such applications is a program registered in the operating system, which allows clients to create and use objects implemented on the server (Elmanova, Trepalin and Tentser, 2003; Identification, n.d.). Another representative of programs with the implementation of a COM server is a multifunctional graphic editor Adobe Photoshop. All the functionality used by the user using the menu, keyboard and toolbar of this editor can be performed automatically, by means of the automation controller. Photoshop gives the automation controller access to functionality through its object model, which is a hierarchy of objects that provide access to other objects.

* Volodymyr Dahl East Ukrainian University, Faculty of Information Technology and Electronics, Department of Programming and Mathematics, Severodonetsk, Ukraine, ratov@snu.edu.ua

COM (Component Object Model – model of multicomponent objects) – one of the basic technologies of the Windows operating system. Moreover, many Windows technologies (Shell, Scripting, HTML) implement their application program interfaces (API) in the form of COM interfaces (Elmanova, Trepalin and Tentser, 2003). The main purpose of COM technology is to provide the ability to export objects, the idea of which is that one module creates an object, and the other uses it by accessing methods or services. COM object is a binary code that performs the desired function and has a software interface. The COM object contains methods that are available to an external application through the COM – interface that allows you to use the COM object.

Because the object model of the Adobe Photoshop multifunctional graphics editor has a tree-like structure topped by an Application object, it can be represented as lists (collections) of objects of the same type that are accessed by index. Individual objects can contain collections, in turn, the elements of the collection are ordinary objects. Access to any object or element of the collection is possible only through the root object Application.

Photoshop gives automation controllers access to functionality through its object model. Therefore, to automate work with the editor, it is necessary to have a program class, the methods of which interact with the object model of the editor.

2. CREATING AN AUTOMATION CONTROLLER

The automation controller (Fig. 1, b), using the developed methods of reading data from the xml format (Fig. 1, a), must connect to the PhotoShop interface and using its API, generate files (Fig. 1, c). In the integrated software development environment Embarcadero RAD Studio on the main form of the future controller program placed the necessary components and buttons (Fig. 1, b). When you click on the buttons "PhotoShop template", "xml-files", "PhotoShop diplomas", the appropriate directories are selected and saved in the operating system registry. Setting (unchecking) the appropriate Visible flags leads to the inclusion and use of additional server functions: background mode, report processing in Excel, saving jpeg-copies.

Fig. 1. Scheme of data flow from the automation controller to the automation server Adobe Photoshop

Selecting the "Compression ratio" component (Fig. 1, b) causes the controller text box to display the percentage of jpeg-copy compression, which affects the quality and size of the resulting jpeg-copy file.

The TController class has been created to control the automation server (Fig. 2). In the private section of the TController class the variable docPS of type Variant is declared. Additionally, event handlers have been created that are associated with components (a reference to the ComObj and Variants modules in the uses section is required).

The TController class introduces a public-variable PhotoShop of the Variant type (Fig. 2, line 8), which is initialized by the CreateOleObject function (Fig. 2, line 11) contained in the ComObj module. When executed, the CreateOleObject function, by calling several COM API functions, will create an instance of the COM object and return its IDispatch interface inside the variant variable. This object contains the interface of the COM object (Adobe Photoshop automation server), the methods of which will be used in the controller program. Considering the implementation of the CreateOleObject function in the source text of the ComObj module, we can find that it, in turn, calls the CoCreateInstance COM API function, which is intended to create an object from an executable file or DLL, ie uses a class factory call. A variable of type Variant can contain various data (string, number, including the interface of the COM object). The Controller object is created by calling the class constructor (Fig. 2, line 10). To save the converted data from an xml file in the TController class, an associative array ArrAttribute with key attributes of xml-files is created (Fig. 2, line 4). To read the xml-file and create an associative array by file attributes, the method of the readXML class was introduced (Fig. 2, line 6). The integrated development environment allows you to access the methods and properties of objects within the variables, and the existence of these methods and properties in the General case may be unknown in advance.

```
1  TController = class()
2  private
3     docPS : Variant;
4     arrAttribute : TStrings;
5     constructor Create;
6     function readXML(fName:string; var countDoc:integer; moveNodes:boolean):TStrings;
7     function PhotoShopOpen(filePattern:string):Variant;
8  public PhotoShop : Variant;
9  end;
10 Controller := TController.Create;
11 Controller.PhotoShop := CreateOleObject('Photoshop.Application');
12 Controller.PhotoShop.visible := not(Form1.CheckBox_FoneMode.Checked);
13 Controller.readXML(FilesXML[i], colDoc, true);
14 if (Form1.CheckBox_PS.Checked AND (findRecordExcel in [NotFind, FindAndDel])) then begin
15  Controller.docPS := Controller.PhotoShopOpen(filePattern);
16  Controller.docPS.LayerSets(1).Layers(1).TextItem.Contents := Controller.arrAttribute.Values[prefics+'DocumentSeries']+
17                  ' № '+Controller.arrAttribute.Values[prefics+'DocumentNumber'];
18  Controller.docPS.LayerSets(2).Layers(1).TextItem.Contents := Controller.arrAttribute.Values[prefics+'FirstName'] +
19                        ' ' + Controller.arrAttribute.Values[prefics+'LastName'];
20  Controller.docPS.SaveAs(directory_of_result_facultet + '\' + ps_file_name);
21  Controller.docPS.Close();
22 end;
```

Fig. 2. Initialization of the TController class and its application for connection to the PhotoShop interface when automating file generation

When creating controllers, the compiler does not check whether an object actually stored in a variable of type Variant, but simply takes the name of the property or method as a regular character string, without analyzing its contents.

To download the document template, a method of the PhotoShopOpen class was created (Fig. 2, line 7), its activation is shown in Fig. 2, line 15. The connection to the Adobe Photoshop automation server in this project is performed dynamically by clicking on the "Perform processing" button. The button works as a switch – if there is no connection to the server, the server starts and a connection is established with it, and if there is - the connection is broken, and the server is unloaded from memory. Code required to disconnect the server: the variant variable is assigned an indefinite value (unassigned).

With this assignment, code is generated that checks what is in the variant variable, and if there is a link to the interface, then its Release method is called. Because the server is called dynamically, each method is checked before each method is called,

Access to the corresponding layers of the PhotoShop template file is due to the use of methods of the COM-server LayerSets, Layers (Fig. 2, lines 16, 18). The data to be downloaded is used from the arrAttribute associative array, which belongs to the Controller object.

At the end of the work, the processed file is saved in psd format (Fig. 2, line 20) and the COM server completes its work (Fig. 2, line 21). When the appropriate check boxes are set, the controller additionally processes the report in Excel and saves a jpeg copy of the processed file. The COM model provides the ability to create components that are reusable and their creation does not depend on the programming language. Such components are called COM servers and are program files (EXE) or dynamic boot libraries (DLLs), specially designed to allow their universal call from any program written in a programming language that supports COM (Elmanova, Trepalin and Tentser, 2003).

3. CONTROLLER AUTHORIZATION

In information systems, authorization (permission, authorization) is understood as granting a certain person or group of persons the right to perform certain actions, as well as the process of verification (confirmation) of these rights when trying to perform these actions (Authorization, n.d.; User authorization, n.d.).

Quite promising for program protection is the method of authorization via the Internet (Varlataya & Shakhanova, 2007), which involves the initial activation of the product on the user's computer, as well as user authorization on the developer's server each time you run the program. This method is not widespread and there are no ready-made solutions in open access (Sklyarov, 2004; Software as a service, n.d.). User authorization at each launch of the program allows the developer to monitor statistics on the use of the program, detect cases of license violations, revoke the licenses of dishonest users, as well as flexibly change the licensing policy (Service Software Activation Service, n.d.).

The article proposes a software authorization control protocol based on the use of binding to the computer hardware and encryption with a 128-bit MD5 public key hashing algorithm (Moldovyan & Moldovyan, 2005; System for software retrieval of additional information from unauthorized copies of Asprotect, n.d.; RSA class, n.d.; MD5, n.d.). The scheme of packet exchange between software with a remote server is presented in Fig. 3.

Consider the authorization algorithm of the automation controller Diplom.exe, which is installed on the user's computer User. The remote server is owned by the controller developers or their trustee.

The software authorization protocol generates a public key (key_client), which will be encrypted with a 128-bit MD5 hashing algorithm. The private key (key_server) must be stored on the http server. The first time you run Diplom.exe, you get the Identification ID from the http server, provided that the administrator has given general permission to work on the system and add new users. Whenever a user tries to perform one of the actions assigned to the developer to control (for example, launching a program, creating, opening, or saving a document), Diplom.exe with the RequestRespons.dll dynamic library connected sends a request to the server to continue by following these steps. data transfer:

1. The automation controller Diplom.exe (1) (Fig. 3) with the help of a dynamic library (2) (Fig. 3) checks the Internet connection and in its absence hides the working functionality of the program issues a message.
2. In the admin object from the classRequest class of the dll library (2) (Fig. 3) the parameter Comp – the name of the local computer and the parameter IP – the IP address of the local computer are formed.
3. In the admin object from the classRequest class of the dll library, the SerNum parameter is calculated – binding to the hardware of the computer where it is installed (4-byte serial number of the system disk C).

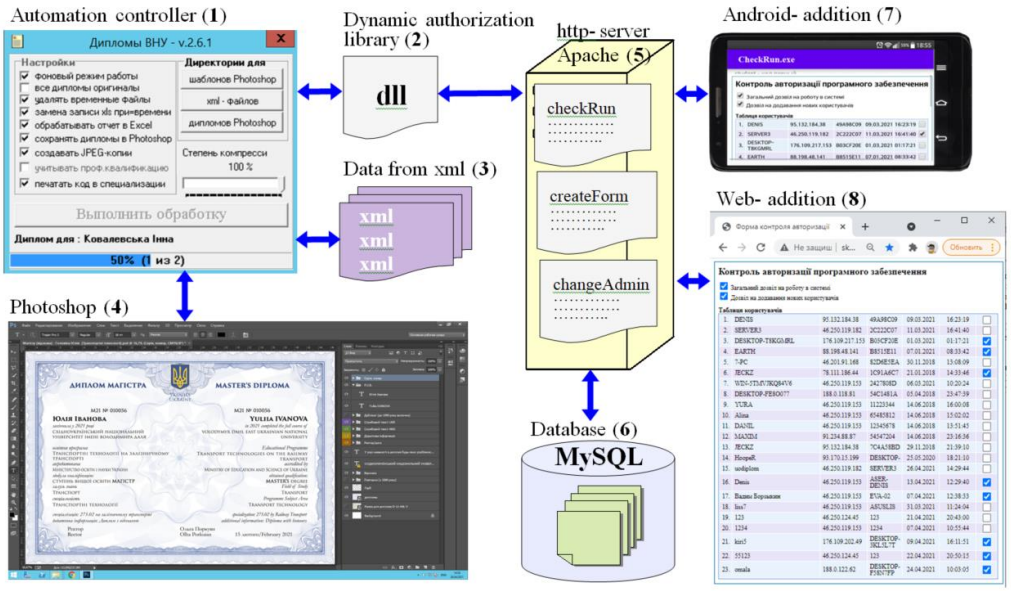

Fig. 3. Scheme of packet exchange between software for which authorization control is organized with a remote server

4. The controller Diplom.exe (1) (fig. 3) by means of dynamic library dll (2) transfers a packet with parameters key_client, Mode, IP, Comp, SerNum to the server (5) (fig. 3).
5. The server (5) checks the possibility of using the user program with the ID Identification, which is determined by the name of the local computer Comp, the IP address of the local computer – IP, the serial number of the system disk SerNum and compares the encrypted key key_client with the key key_server.
6. The server (5) sends a packet with the parameter f_status of the admin object from the classRequest class of the dll library associated with the Diplom.exe controller.
7. The controller (1) (Fig. 3) using a dynamic library (2) (Fig. 3) receives a packet from the server (5) (Fig. 3). If the f_status parameter confirms the ability to run, the dll library (2) (Fig. 3) activates the main form of the program (1) and displays the work panel, and if not, it issues a message and terminates the controller.

In the integrated development environment Visual Studio developed a dynamic library that integrates with the automation controller Diplom.exe and controls its authorization, while implementing early binding (early binding) of the client to the server, ie the creation and use of the controller table of virtual methods tables of virtual server methods.

The interface of the classRequest class, the methods of which implement the controller authorization protocol, is shown in Fig. 4. A 128-bit MD5 hashing algorithm was chosen as the encryption mechanism (MD5, n.d.).

The classRequest class (Fig. 4) consists of:

1. Private URL field, which specifies the address of the web-resource for pinging the presence of a connection to the global network.
2. Private structure locAnsver in which the answer from the server is formed.
3. Private Host method, in which the resource host is calculated from the resource address.
4. Class designer with pre-initialization.
5. Common methods GetPCName, GetSerialNum, which return the name and serial number of the computer.
6. The public method isInternetConected, which returns the sign of connection to the global network.
7. A public Query method that executes a GET request to a remote server.

```
namespace Control
{
    public struct AnsverServer  {...}
    public class classRequest
    {
        private string URL;
        private AnsverServer locAnsver;
        public classRequest(string url = "http://www.google.com") { this.URL = url; }
        private string Host(string URL) {...}
        public virtual AnsverServer ansver { get { return locAnsver; } }
        public virtual string GetPCName { get { return Environment.UserName; } }
        public virtual string GetSerialNum { get { return Environment.MachineName; } }
        public virtual bool isInternetConected {...}
        public virtual void Query(string addUrl) {...}
    }
}
```

Fig. 4. ClassRequest class

According to the developed protocol of authorization of the controller, for use of functionality of dynamic library dll in the project of the controller Diplom.exe it is necessary to change the work scenario (fig. 5). Instead of initializing the main form of the program (1), an instance of the class classRequest – admin (2) is created to ping the presence of a connection to the global network (3) and to send and receive packets from a remote server (4).

The created object allows one of three possible scenarios of the further work of the program:

1. Upon receipt of the activation permission package, the Diplom.exe controller will report the successful authorization (5) and perform the standard work of the program (Fig. 5).
2. In the absence of the Internet, the Diplom.exe controller will notify (7) and deactivate the possibility of standard operation.
3. Upon receipt of the prohibited activation package, the controller Diplom.exe will generate a message from the server (6) and deactivate the possibility of standard operation.

The server part of the software package is implemented in the PHP programming language with the ability to query the MySQL database server.

```
using System;
using System.Windows.Forms;
using Control;
namespace Check_Run
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            //Application.Run(new Form1());

            classRequest admin = new classRequest();
            if (admin.isInternetConected) {
                admin.Query("http://.../server_script/checkRunProgramms.php?" +
                            "mode=checkRun" + "&comp="    + admin.GetPCName.ToString() +
                                              "&serNum=" + admin.GetSerialNum.ToString());
                if (admin.ansver.Status == 1) { Application.Run(new Form1()); }
                else { MessageBox.Show(admin.ansver.Message, "Ответ от сервера",
                                       MessageBoxButtons.OK, MessageBoxIcon.Information, 0);
                }
            }
            else {MessageBox.Show("No internet connection" + Char.ConvertFromUtf32(10) +
                                  "It is not possible to authorize", "Operation error",
                                   MessageBoxButtons.OK, MessageBoxIcon.Error, 0);
            }

        }
    }
}
```

1 2 3 4 5 6 7

Fig. 5. Implementation of the authorization mechanism in the Diplom.exe controller project

An embedded classRequest object can communicate with a server over the World Wide Web by its domain name or IP address.

Permission to activate the program is delegated by the remote server and can be based on the following data: the number of successful launches of the controller, the time of use of the program, the number of files created when using the controller and more.

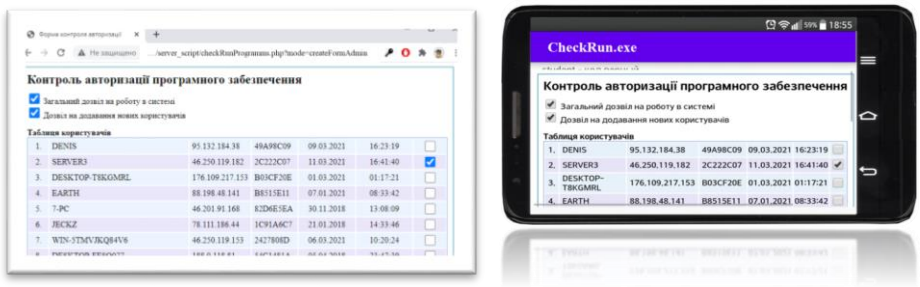

Fig. 6. Web-form and android-application of software authorization administration

Before the first use, the Diplom.exe controller receives an ID and, if the administrator has given general permission to work in the system and add new users, activates its functionality. Using the web-form of administration of the authorization of the controller or android-application (Fig. 6), the system administrator can control and monitor the user's

work with Diplom.exe. This process is carried out by adjusting the list of features and functionality of the controller for individual users or a general ban on activation (Fig. 6). For unique registration data for this identifier, the MySQL database stores the name of the local computer, the IP address of the local computer and the 4-byte serial number of the system disk of the computer on which the automation controller was involved.

4. CONCLUSIONS

As part of the study, a class was developed that provides control of the internal process automation server. This server performs its actions in the address space of the graphics editor Adobe Photoshop. This allowed to create a controller that automates the formation of image files.

Using the software authorization protocol, a dynamic library was also developed that integrates with the automation controller and monitors its authorization and functionality. For the PHP hypertext preprocessor running on the Apache http server, scripts for the interaction of the dynamic library and the MySQL database have been developed, a script for creating a web form of administration has been written, and a controller authorization administration application has been created in the Android Studio integrated development environment.

The obtained software controller and means of protection against unauthorized use and control of authorization became the basis of the software package "Diplomas SNU v.2.6.1" at the Volodymyr Dahl East Ukrainian National University. This complex is designed to create a single register of diplomas at the university, to automate the creation of files-diplomas of higher education in the multifunctional graphic editor Adobe Photoshop. All data for the analysis and formation of diplomas the controller exports from parameters of the corresponding xml-files downloaded from the uniform state base of education.

Authorization of the user at each start of a software complex allows the administrator to watch statistics of use of a complex, and also provides its protection against unauthorized use.

Scientific novelty is to obtain a class with methods that provide access to the functionality of the graphics editor using its object model and create a means of software protection against unauthorized use and control of the authorization of the software package. The practical significance lies in organization of software interaction with the COM interface of the multifunctional graphic editor Adobe Photoshop to automate the process of forming documents with restriction of unauthorized access.

REFERENCES

Artemov, M. (2007). *Basics of COM technologies.* Voronezh. p. 84.

Authorization. (n.d.). *Wikipedia.* Retrieved May 12, 2021 from https://ru.wikipedia.org/wiki/Authorization

Elmanova, N., Trepalin, S., & Tentser, A. (2003). *Delphi and COM technology.* Petersburg. p. 698.

Identification. (n.d.). *Wikipedia.* Retrieved May 12, 2021 from https://ru.wikipedia.org/wiki/Identification

MD5. (n.d.). *Wikipedia.* Retrieved May 12, 2021 from https://ru.wikipedia.org/wiki/MD5

Moldovyan, N., & Moldovyan, A. (2005). *Introduced at the cryptosystem with an open key*. Petersburg. p. 288.

Ratov, D. (2020a). Architectural paradigm of the interactive interface module in the cloud technology model. *Applied Computer Science, 16*(4), 48–55. http://doi.org/10.23743/acs-2020-28

Ratov, D. (2020b). Model of the module for the interface of the information web-system. *Mathematical machines and systems. Kiev, 4,* 74–81.

RSA class. (n.d.). *Code.google.com.* Retrieved May 12, 2021 from http://code.google.com/p/phpjsrsa/source/browse/trunk/rsa.class.php?r=6

Service Software Activation Service. (n.d.). *Softactivation.com*. Retrieved May 12, 2021 from http://www.softactivation.com/asp/getstarted.asp

Sklyarov, D. (2004). *Mystery for the capture of evil information*. Petersburg. p. 288.

Software as a service. (n.d.). *Wikipedia*. Retrieved May 12, 2021 from http://ru.wikipedia.org/wiki/Software_as_a_service

System for software retrieval of additional information from unauthorized copies of Asprotect. (n.d.). *Jak.koshachek.com*. Retrieved May 12, 2021 from https://jak.koshachek.com/articles/asprotect-32-sistema-programnogo-zahistu-32-bitnih.html

User authorization. (n.d.). *Academic.ru.* Retrieved May 12, 2021 from https://dic.academic.ru/dic.nsf/business/17457

Varlataya, S., & Shakhanova, M. (2007). *Software and hardware protection of information.* Vladivostok. p. 318.