NTNU
Innovation and Creativity

KOPEK Payment System as a Licensing Solution for Software

Richard Husevåg

Master of Science in Computer Science

Submission date:	September 2006
Supervisor:	Bjørn Olstad, IDI
Co-supervisor:	Stein Hardy Danielsen, KOPEK AS

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

The candidate will examine how the KOPEK Payment System can be used as a licensing solution for commercial software. A market analysis of existing solutions should be included in the report, as well as suggested improvements that would make the KOPEK Payment System better suited to this purpose. A prototype that utilizes the KOPEK Payment System as a licensing solution should be implemented.

Assignment given: 15. May 2006
Supervisor: Bjørn Olstad, IDI

Acknowledgements

Without the input and background material provided by Øivind H. Danielsen and Stein H. Danielsen this report could not have been written. I would like to thank them for their support as well as for all their work in developing the KOPEK Payment System upon which this report is based. Particularly Stein H. Danielsen has provided me with good help and input during the work on this report. Thanks also to my tutor from the Norwegian University of Science and Technology, Bjørn Olstad.

The faculty of Information Technology, Mathematics and Electrical Engineering at the Norwegian University of Science and Technology should know that I am grateful that they approved my writing of this thesis.

Table of contents

ACKNOWLEDGEMENTS I

TABLE OF CONTENTS II

PREFACE V

1 INTRODUCTION 1

2 THEORY AND BACKGROUND 2

2.1 KOPEK AS 2
2.2 KOPEK PAYMENT SYSTEM 3
2.2.1 KOPEK Payment Server 3
2.2.2 KOPEK Client 4
2.3 ENCRYPTION AND SECURITY 5
2.3.1 Encryption Keys 5
2.3.2 Shared Key Algorithms 5
2.3.3 Public Key Algorithms 5
2.4 IDENTIFICATION CERTAINTY BY AUTHENTICATION 6
2.4.1 Public Key Infrastructure (PKI) 6
2.5 PRIVACY 7
2.6 RESELLER, DISTRIBUTOR AND MANUFACTURER RELATIONS 8
2.6.1 Customer Ownership 9
2.7 REVENUE SHARING 10
2.8 KOPEK CLIENT SOFTWARE 11
2.9 MICRO-PAYMENT 12
2.10 LICENSING 13
2.10.1 License agreement 13
2.10.2 Licensing system 13
2.10.3 Licensing models 13
2.10.3.1 Time-limited license 14
2.10.3.2 Module based license 14
2.10.3.3 Demo/Trial license 15
2.10.3.4 Personal license 15
2.10.3.5 Single computer license 16
2.10.3.6 Per user licensing 16
2.10.3.7 Per client licensing 16
2.10.3.8 Per processor licensing (per server licensing) 17
2.10.3.9 Site license 17
2.10.3.10 Academic/discount licensing and licensing programs 18
2.10.3.11 Rented licensing 18
2.10.3.12 GNU General Public License/Open source licenses 18
2.11 ERP- AND CRM-SYSTEMS 19
2.12 SUMMARY 20

3 MARKET ANALYSIS / STATE OF THE ART 21

3.1 INTRODUCTION 21
3.2 MACROVISION 21
3.2.1 Macrovision FLEXnet Publisher 21
3.3 SAFENET 23
3.3.1 Sentinel RMS 23
3.4 SWREG 24
3.5 SUMMARY 25

4 USING THE KOPEK PAYMENT SYSTEM AS IS 26

4.1 INTRODUCTION 26
4.2 OVERVIEW OF THE KOPEK PAYMENT SYSTEM 27
4.2.1 Overview of the KOPEK Payment System basic design 27
4.2.2 KOPEK Payment Server configurations 28
4.2.3 Overview of the KOPEK Payment System from an end user's perspective 29
4.2.4 The KOPEK Payment Client account operations 32
4.3 INTERNET COMMUNICATION CLASS 34
4.4 HOW THE KOPEK PAYMENT SYSTEM WORKS 38
4.4.1 Internet web server request 38
4.4.2 License validation procedure 38
4.4.3 KOPEK Payment Server configuration 38
4.4.4 Handling different licensing models 40
4.4.4.1 Time-limited license 40
4.4.4.2 Demo/trial license 41
4.4.4.3 Personal license 42
4.4.4.4 Single computer license 42
4.4.4.5 Per user license 44
4.4.4.6 Per client license 44
4.4.4.7 Per processor license 45
4.4.4.8 Academic/discount license and license programs 46
4.4.4.9 Rented license 46
4.4.4.10 GNU General Public License (GPL)/Open license 46
4.5 HARDWARE KEY 46
4.6 PAYMENT CONSIDERATIONS 47
4.7 KOPEK PAYMENT SYSTEM VS. OTHER LICENSING SYSTEMS 48
4.8 SUMMARY 49

5 RELIABILITY AND STABILITY ISSUES 50

5.1 INTRODUCTION 50
5.2 CONSEQUENCES IF KOPEK AS GOES OUT OF BUSINESS 50
5.3 CONSEQUENCES IF THE SOFTWARE PUBLISHER GOES OUT OF BUSINESS 50
5.4 INTERNET CONNECTION FAILURE 50
5.5 SUMMARY 51

6 SUGGESTED IMPROVEMENTS AND PROTOTYPE IMPLEMENTATION 52

6.1 SUGGESTED IMPROVEMENTS 52
6.1.1 Support for more license models 52
6.1.2 ERP- and/or CRM-system integration 52
6.1.3 Reliability improvements 53
6.2 PROTOTYPE IMPLEMENTATION 54
6.2.1 Web form user registration and license information recording 55
6.2.2 Software application prototype 62
6.3 SUMMARY 67

7 SUMMARY AND CONCLUSION 68

7.1 SUMMARY 68
7.2 CONCLUSION 69

APPENDIX A. TABLE OF FIGURES 70
APPENDIX B. TABLES 70
APPENDIX C. TABLE OF REFERENCES 71
APPENDIX D. MYSQL DATABASE DEFINITION FOR SAMPLE DATABASE 72
APPENDIX E. KOPEK PROTOTYPE APPLICATION SOURCE CODE 73

Preface

This report will investigate what is required from a licensing solution for commercial software and evaluate whether the KOPEK Payment System is applicable to this purpose. An attempt will be made to make a prototype of a system based on the KOPEK Payment System that is suited to the purpose of license control.

1 Introduction

The first part of the report will provide sufficient background material to enable the reader to understand what is dealt with in the rest of the report. Then existing solutions already on the market for license control will be analysed and the requirements for such solutions will be presented. Since the purpose of this report is to try to use the KOPEK Payment System as a basis for a similar solution, the existing possibilities in the KOPEK Payment System will be analysed and finally there will be an implementation of a prototype with the intention of making the KOPEK Payment System better suited to this purpose.

2 Theory and background

2.1 KOPEK AS

Øivind H. Danielsen and Stein H. Danielsen founded the company the spring of 2002. The idea behind the company was based on their vision to make Internet commerce easy and profitable, after having experienced that existing solutions for this were far from optimal. Since they came up with a good theory on how an electronic payment system could be built that would be as easy as possible to use and implement, but still very secure, they started developing prototypes and soon realized their theory would be possible to implement. The idea was that this should not only be easy to use for the end user, but also for the provider of the product. They came up with a concept they called the KOPEK Payment System.

2.2 KOPEK Payment System

The KOPEK Payment System is designed primarily with payment of online content in mind. Originally the idea was to conquer the emerging market for so called micro-payment. That required a very short path for the end user (customer) from seeing the product on the Internet to purchasing it and making the payment as quickly as possible whilst still preserving security. Since the margins in this market are very small it also required an extremely cost efficient way for the providers of the products to make their products available in the system. So the KOPEK Payment Server was designed to cover the requirements of the providers and the KOPEK client was made to ensure security and ease of use for the end users (customers).

2.2.1 KOPEK Payment Server

Since the easiest way of explaining how the KOPEK Payment Server works is through examples of web pages with content that a provider will require payment for, this is used here. But it is in no way limited to this purpose only.

The provider of a product to be sold on the Internet will always need to generate a web site with the product itself or a description of the product on. But since a payment system often has been required to be implemented as part of the web pages source code it has not been possible to just create clean web pages when payment will be required. Another consequence of such an implementation is that the choice of payment system often gets fixed to the payment system first chosen since it is quite an effort to code the system into the web pages source code. The cost of changing system will simply be to high for this to be an option.

The KOPEK Payment Server however is placed in front of an existing web site and then all the content of an existing web site are filtered through the KOPEK Payment Server. That makes it easy to implement on any existing web site or any new web site. The products to be sold are defined using filters on the content of the web site. So in reality it is an access control system for one or more web sites. If a filter on a web page has been set in the KOPEK Payment Server that page will not be available for any Internet user unless the requirements set in the KOPEK Payment Server for the page are met. The requirements can just be that a login is required for access or that payment is required before access is granted. Or there can be no requirements at all if it is a page that should publicly available.

2.2.2 KOPEK Client

The best way to ensure a totally secure and encrypted connection to the correct KOPEK Payment Server all the way from the end user (customer) is to have a local client application installed on the end users computer. This also has other advantages, like the fact that it enables true single sign in. The client application can still be left running even if a browser is closed or another application is closed. The user's login to the KOPEK Client can then be used by any number of web sites or applications for access control, provided that these have implemented support for the KOPEK Payment System. The level of identification certainty required (see chapter 2.4) can be selected by the web site or application in question. Another advantage of using a client application like the KOPEK Client for single sign in is that an HTML based authentication can be avoided. HTML based authentications used for single sign in are vulnerable in the sense that an HTML web page can be easily duplicated and a malicious part could claim to support the HTML based single sing in for some services or a service. Hence this malicious part could easily obtain the username and password of the users who are conned into using these services or this service.

2.3 Encryption and Security

Encryption is a way to make any kind of data difficult to read for an unintended reader. By using encryption the data can be considered more securely transferred than it would otherwise be. The encryption method used will determine the level of security that can be achieved. Any kind of data, like computer data traffic or human voice traffic could be encrypted and in many cases this is common today. The user isn't necessarily affected in any way, but if the system should ensure that only the intended parts participate in the communication the participants at least have to identify themselves securely. How this is done will be discussed in the paragraph *'Identification certainty'* later in this chapter. The following summarizes the most common and secure encryption algorithms used today (background material in R.1, chapter 7.1).

2.3.1 Encryption Keys

When encryption is done today it usually involves a code or a key that has to be used by both the sender and the receiver of the message. Keys are typically numbers, letters, words or a combination of these that are used in an encryption algorithm to make the encrypted message a lot harder to break. If the key is longer it would take a brute force search for the key longer to complete and hence the encryption could be considered more secure. With the use of keys a publicly known algorithm for encryption can safely be used since the security is actually the key and not the algorithm itself. This makes it possible for everyone to encrypt messages strongly, without having to invest huge amounts of money and time in figuring out a new algorithm first. There are several algorithms for this publicly available.

2.3.2 Shared Key Algorithms

Shared-key algorithms depend on the sender and receiver of a message to possess the same key. This however can be difficult to achieve since the sender and the receiver would have to be sure that they really exchange keys with each other and not someone else.

2.3.3 Public Key Algorithms

Some algorithms allow one key to be used for decrypting a message and a different key to be used for the encryption. That enables a communicator to leave his encryption key publicly known (the 'public key') and keep the decryption key secret (the 'private key'). The communicator can then at least be sure that the messages received and which can be decrypted with the private key was intended for the communicator and not anyone else. The challenge with these algorithms is making sure that the public key obtained really is the public key of the intended receiver and not someone else pretending to be him or her.

2.4 Identification certainty by authentication

How can anyone really be sure that they are exchanging information with the right person? In our society we rely on identification tokens like a passport, a drivers license or similar publicly acknowledged identification tokens issued by the government the person belongs to. The token is usually required to at least consist of a picture of the person as well as the key information about the person (name, date of birth etc.) and a unique number that can be used to find the same key information from the publisher of the token.

But such tokens are unfortunately possible to fake, although different techniques are widely used to make faking more difficult. Lately electronic identification tokens have been developed, so that the unique number of the identification token is used to retrieve the key information of the person from the publisher of the token as the token is used for identification. This information can of course still pretty much match a person pretending to be someone else. But it can be made very reliable by also having biometric information stored and possible to verify from the publisher of the token. This however raises many ethic and moral questions, so not everyone would offer this kind of security when issuing tokens. And even with this level of secure identification, the security wouldn't be stronger than the level of security and trust that the publisher of the token supplies.

Another factor that would compromise the certainty of the identification no matter how strong the security level is or what type of identification method is used, is the person that is to be identified. If that person for some reason agrees to let another person use his or her identification tokens or identifies on behalf of another person, then the identification could be considered compromised. It is difficult to know if a person identifying electronically is being threatened into doing so or if the person is letting someone else use the identity voluntarily.

2.4.1 Public Key Infrastructure (PKI)

A commonly used algorithm for electronic identification is called the Public Key Infrastructure. This is algorithm consist of a private-public key pair explained in chapter 2.3.3 and in addition an electronic certificate which matches the public key and identifies the person behind the public key. The certificate consists of the person's personal data and is signed by a PKI-authority provider.

2.5 Privacy

To most people privacy is required and expected when using identification or whenever storage of personal information is done. This means that if someone is trusted to receive personal information like a name, an address or a credit card number they are expected and often by law required to keep this information safely and not let anyone else use it for other purposes than those intended by the trusting person.

A big problem for many Internet users is that they have to leave such personal information at Internet shops they can't be sure to trust. Of course an Internet shop has to know that the user visiting is paying using his own money and that the source of the money has full coverage for the amount required. Since abuse of personal information has been widespread on the Internet, trusting information like this is probably often a limitation for Internet sales. If the information given out could be assured permanently erased after the required use or at least watermarked in some way so that it could be traced, abuse could probably be much more limited. But the best solution would naturally be if the information hadn't been given out in at all.

This is what the KOPEK Payment System is all about. Only the necessary information is given once when registering a new KOPEK account, then this information stays with KOPEK and the KOPEK user identification is the only information that can be shared with the hosting site when shopping at sites using the KOPEK Payment System. The KOPEK Payment System verifies that the customer has coverage for the required amount of money to buy whatever is to be purchased. Any credit card information or other information about the customer is never shared with the site that a product is bought from.

2.6 Reseller, distributor and manufacturer relations

Many products today are produced or assembled at one or more locations of a manufacturing organization. Traditionally, this organization is optimised and focuses on the processes surrounding the production of products. As the products sales increase and the market geographically expands, the organization often realizes that it is more efficient to use local organizations that already knows local culture and regulations to handle sales in a smaller geographically limited market. Then these local organizations are referred to as resellers and they are usually independent organizations.

This also distances the manufacturing organization from the end customer of the product. The manufacturer may even terminate both sales and support to end customers. This is left to the reseller and the authorized resellers will be the only way through which an end customer can receive support and buy the products from this manufacturer. This makes the manufacturing organization able to be more efficient and respond faster to more customers needing support, since it only has to communicate with a limited number of customers; the resellers.

In an even larger geographical market yet another level will often be practical. The manufacturer will then only deal directly with large volume distributors who in turn only deal with resellers. The resellers of course deal directly with the customers and in the other end the distributor.

There are different advantages and disadvantages with more levels of distribution of manufactured goods for the manufacturer. An advantage is that focus can be maintained on research and development as well as the manufacturing process itself. But the disadvantage is that the distance to the end customer of the product becomes longer and that can make it hard to make the correct improvements on the product that are wanted by the market. Also sales may drop as the quality and response time of the customer support may not be good enough from all resellers. In addition, more levels of distribution also imply higher distribution costs, as every level will require profit from the distribution.

2.6.1 Customer Ownership

A company would consider itself to be the 'owner' of its customers or at least its customer portfolio. Still the customer can't be forced to continue shopping from it. So customer ownership isn't related to the customer's rights or obligations. If the customer wants to he or she could even shop directly from a distributor or manufacturer if the distributor or manufacturer allows it. But this is where any reseller would feel betrayed by the distributor or the manufacturer, since the reseller expects some loyalty in return for making the initial sales to the customer. Unless the business model the distributor or manufacturer officially has is based on direct sales as well as reseller sales.

So customer ownership isn't necessarily a legally valid term, but some distributors or manufacturers may want their resellers to feel that they own their own customers in order to limit competition amongst their own resellers. Usually a reseller also has to commit to not selling competing products to the supplier or manufacturer's product if this kind of customer ownership deal is used. That weakens the reseller's possibilities to negotiate terms after the initial negotiation with the distributor or manufacturer, since the process of changing product to a competing product would be harder for the reseller. Since the reseller also probably wouldn't be eligible to supplying the existing customers on the product that is to be abandoned any support or upgrades, all the resellers customers would eventually also have to change product (or change reseller).

The advantage of customer ownership from the reseller's point of view is that customers have to change their product in order to change reseller. The reseller is also likely to get a share of any further income made from selling upgrades or additions to the product. It is also an advantage for a reseller to not have to compete against other resellers of the same product, but this requires that each reseller be assigned its own region to which sales can be made. Customer ownership however can ensure that any sale after the initial sale of products from the distributor or manufacturer in question will go through the reseller that made the initial sale.

For resellers this kind of ownership can be achieved through the licensing model used. The reseller can be a part of the license agreement signed by the customer in such a way that all contact between the customer and the distributor or manufacturer must go through the reseller or possibly another reseller.

But lately it has been more common to deploy electronic license agreements where the reseller isn't included at all. The customer only accepts the license agreement during an installation procedure or during the first time deployment. If the reseller cannot be included in the license agreement itself, the reseller probably has to make sure solid agreements with the supplier are made prior to selling products from the supplier so such interests are preserved.

Using the KOPEK Payment System will not affect the contents of a software publisher's license agreements, the software publisher can choose to use electronic license agreements or even use customer signed license agreement papers. These license agreements in addition to the reseller's agreements with the software publisher will determine how customer ownership is and should be handled. The issue with using an electronic payment system for payment of licenses however can violate the reseller's customer ownership if the payment has to be done directly to the software publisher and not go through the reseller. With the KOPEK Payment System it should be possible to preserve that payment has to go through the reseller.

2.7 Revenue Sharing

Any reseller expects a part of the revenue made when selling a product. Otherwise the reasons for selling would be much more limited. What kind of revenue the reseller is given will vary from reseller to reseller. Usually a fixed percentage of the recommended sales price, which is supplied by the manufacturer, is used. Traditionally it has been possible for resellers to set their profit themselves by raising or lowering the price of the products. Since the customer is billed by the reseller directly this has been easy. But there are issues with this model of sales. Often these issues are solved by competition amongst resellers or limitations given by the manufacturer.

With the KOPEK Payment System however, the billing can be done directly from the manufacturer to the customer and not via the reseller at all. The reseller's share of the revenue can be set as a percentage in the KOPEK Payment Server. Therefore the manufacturer has to adjust the percentage the reseller will be granted if it is to be changed. This eliminates the reseller's opportunity to bargain a better price with the supplier without passing this on to the customer, and it limits the reseller's possibilities to charge customers more than the manufacturer would want them to. A reputation of being too expensive would probably not be positive for the manufacturer.

If a centrally controlled billing isn't used, it will be possible for a reseller to cheat the supplier for revenue in this way. The supplier will also realize this possibility and drive a harder bargain, so it could perhaps in some cases be more difficult to deliver the correct price to the customer. Sales could be lost due to distrust between reseller and software publisher.

2.8 KOPEK Client Software

The KOPEK client software consists of an application that can be downloaded and installed independently of the KOPEK server software. This application can be used independently as well, since it includes features for instant messaging systems. It supports commercial instant messaging protocols like Microsoft Messenger, AOL Instant Messenger, ICQ and the Jabber protocol. All with limited functionality. In addition it includes its own KOPEK instant messaging protocol, which ensures a fully encrypted communication. The KOPEK instant messaging protocol also support web camera pictures. Of course the main focus for the client software is the payment system. The client gives the user full control of all transactions made and possibility to manage his or her own KOPEK bank account. Whether the user wants to use the KOPEK bank account at all or simply use the other payment methods available (like a credit card) is of course up to the user.

2.9 Micro-Payment

Micro-payment is a term used for transactions of very low value. The idea with micro-payment is that it enables an organization to charge a customer an almost insignificant amount of money for a single purchase. The customer however will usually need to or want to make more than one purchase. Since the price is so low, the customer can be persuaded into making many purchases more easily. And if a company selling a product can get many customers to spend lots of small amounts on very many purchases without own expenses connected to each purchase, the company can make this profitable. The advantage from the customer's point of view is that they only need to buy the product they really want - not a much more expensive package of products they don't need, but which also includes the product they wanted.

The principle with micro-payment is commonly that a customer is charged a reasonable amount of money, which is put on an internal account for that customer. That account is then charged with the smaller amounts each purchase represents until it needs to be refilled. This is just like any debit card, except that the amounts used are smaller. A good example of how this works can be found at the Russian online music stores http://www.allofmp3.com and http://www.alltunes.com.

Still these solutions would be better for the customer if the same account could be used for purchases on many other stores as well. As long as all the deposited money has to be spent in the same store, a customer only wanting to make a single purchase once might as well buy a package for the same amount as the minimum amount it is possible to deposit on this account since the money are in fact lost for the customer if the customer never returns after the first purchase. Of course this may be a nice way for the store to make more customers return.

Although micro-payment is all about very low value transactions, there are limits to how low this value should be. In order for the customer to do a transaction, the customer has to spend some time and make an effort. This is an expense even though it isn't always directly connected to cost in money. But if the product being sold has a too low monetary value, the customer typically would feel the effort of retrieving the product was too high. In other words, the lower the monetary value of a product, the lower the effort a customer has to put in has to be. Hence there will always be a practical lower limit to the value of the products sold in order to make sales.

2.10 Licensing

2.10.1 License agreement

Software purchased today is usually a subject to some sort of license agreement between the software publisher and the end user or the end user's company (the customer). These agreements usually set conditions for the use of the software and include disclaimers to protect the software publisher from responsibility for consequences of use. In addition the license agreement can contain warranties and responsibility the software publisher undertakes by making the software available for the customer.

2.10.2 Licensing system

To make sure that the end user doesn't violate the license agreement so that it decreases the publisher's incomes, the publisher could incorporate a licensing system in the software. The licensing system could also be referred to as a license control system. Since the most apparent way of violating a license agreement for software would be to distribute copies of the software that aren't legally purchased, the main focus for such systems is to prevent this. But very few if any systems that exist today can guarantee that the license agreement isn't violated. But such a system usually reduces the chance of this happening, and depending on the system the chance can be reduced to an acceptable level for most publishers. There is always the challenge of usability versus security when implementing such systems, so compromises often have to be made on both of these in order to make a system customers will accept.

2.10.3 Licensing models

Typical licensing models commonly used are the following:

- Time-limited license
- Module based license
- Demo/trial license
- Personal license (one license for one user)
- Single computer license (one license for one computer)
- Per user license (every single user of the multi-user software must have a license)
- Per client license (maximum simultaneous users of an application)
- Per processor license
- Site license
- Academic license and other discounted varieties of licenses (license programs etc.)
- Rented license
- GNU General Public License (GNU GPL)/Open Source

2.10.3.1 Time-limited license

Time-limited licenses exist in all license models mentioned in chapter 2.10.3 and they very often offer the possibility of renewal when the time limitation period expires. There are some security considerations regarding how to ensure that an expired license is renewed, which also apply to all license models, but they are most obvious when issuing time limited demo/trial licenses and these issues are hence discussed in chapter 2.10.3.1.

2.10.3.2 Module based license

Each of the license types mentioned in chapter 2.10.3 can also be for a full access to an application or just for a part of an application (referred to as software module). The license is referred to as module based if there are different licenses for the different software modules. Module based licenses can have different time-limitations for each module or all modules can have the same time-limitation (all modules expire at the same date and time).

2.10.3.3 Demo/Trial license

The demo or trial license is probably more suitable for limited access to software functionality than full access to the software for a limited period of time. This is due to the fact that a customer could just renew the software once the license expires, and then just renew the demo/trial license, which is free. One could argue that time limited editions would be desirable, but that each single person only could gain access to the time limited edition once. However, the fact that a physical person can easily pretend to be a new person once the period of time expires makes this goal hard to achieve. In addition time limited editions have historically had the disadvantage of being exploited simply by customers adjusting their computer's clock. This because the time expiration control often only is performed when the application starts and so the clock can be re-adjusted after the application has been started. By using a server on the Internet as the time source, the customers would not be able to override the time limit by adjusting their own computers clock. And the application could of course be built so it performs the time expiration control regularly. But only allowing a physical person to retrieve a time limited license once would require that the person is identified securely and not just by a user created without any identification. The issues of identification certainty are further discussed in chapter 2.4.

2.10.3.4 Personal license

Software publishers of commercial software very commonly issue a personal license. This license is for any private person or wherever a company wants to buy a cheaper license for a single employee. Since the license is personal, this represents a disadvantage for a company in the sense that the license usually cannot be transferred from an employee leaving the company to a new employee. Exceptions from this occur, mainly from software publishers that don't issue other types of licenses. Personal licenses can also be restricted in functionality if they are significantly cheaper than other license types available, but personal licenses that have the more advanced functionality may still exist at a higher price if a market for such licenses exist.

2.10.3.5 Single computer license

This is probably one of the most commonly used licensing models by applications that are copy protected. Despite this there are several legal aspects of it that has to be considered when using it. Doesn’t a customer have the right to start an application on another computer if the computer for which the license was bought fails? Even if the computer doesn’t fail, shouldn’t the customer be able to switch computers at whatever time? The answer to these questions is probably a definite ‘yes’ from most software publishers. The intention with a single computer license is usually to allow more than just the purchaser’s user to run the software, but not more than one user simultaneously. It is not meant to restrict the software to only be run on one specific computer, but since the traditional way of using computers has been that only one user can use a physical computer at a time, restricting the license to this computer has been an acceptable way of licensing software. Different physical users can use the same computer and many software publishers find it reasonable that one single license should cover all these users. But this causes a conflict with the question of allowing the customer to switch computer. How can it be assured that the software isn’t distributed to several computers and used simultaneously by more than one user if the software isn’t limited to a specific computer only? In addition the introduction of multi-user session operating systems has rendered this licensing model questionable. When one computer can have an almost unlimited number of user sessions simultaneously connected and thereby run the software in several simultaneous instances the intentions of only allowing one simultaneous user run the software could be violated if the only restriction is which physical computer it can run on. So the new licensing model taking over for the single computer license would be the licensing model that makes the customer buy a number of licenses corresponding to the number of simultaneous users using the software (per client licensing).

2.10.3.6 Per user licensing

This licensing model is in many ways similar to the personal license model. The difference is that when licenses for different components of a software has been purchased for one person, the next person who needs access to the same software has to pay a smaller fee than the first user to gain access to all the already purchased components of the software. Of course, this only applies to users within the same organization. In other words, this is a discount system for additional personal licenses. Very often there is a mandatory and more expensive server license (see chapter 2.10.3.8) that has to be purchased before the user licenses can be bought.

2.10.3.7 Per client licensing

Per client licensing will, like per user licensing, favour more than one user of the software within the same company. This one only limits the number of users that can use the software simultaneously, so the licenses are not personal but can be used in turn by all users.

2.10.3.8 Per processor licensing (per server licensing)

Software that is used as server software for a potential unlimited amount of simultaneous users, like Internet server software, can be unpractical to license by the number of simultaneous connections or distinct persons using it. Therefore a licensing model exists which only limits the number of processors the software can be run on simultaneously. The price for such licenses will often be much more expensive than the other licensing types since the number of simultaneous client users connected to the server software is unlimited. This license model is in many ways just a variant of the site license discussed in the next chapter, but since it is easier to electronically verify that a license isn't abused when limited to certain pieces of hardware than it is when limited to certain human beings a separate chapter was written for this license model. Variants of this license model also exist, limiting the license to other hardware components that are more or less suitable for the purpose of restricting the use of the license.

2.10.3.9 Site license

A site license agreement is usually issued to a large organization or parts of it for the purpose of making license control management easier for that customer. These licenses are often very expensive, but the customer in turn has an unlimited amount of licenses that can be used by all the parts of the customers organization included in the site license agreement. This could be geographically limited, limited to only certain types of employees, employees of certain parts of the organization and so on. A site license is usually negotiated individually for each customer, so which limitations that apply in each case will vary a lot. The point is that for a large organization it will often be more cost efficient to pay a software provider a larger amount of money in order to know that all their software licenses are legal, than it is to administer a fixed amount of licenses as the organization grows.

2.10.3.10 Academic/discount licensing and licensing programs

These licenses are just discounted versions of the license types mentioned above. But any software licensing system should preferably also support having discounted versions of the licensing model or models the software is available in.

2.10.3.11 Rented licensing

By rented licensing in this report, it is referred to licenses owned by for instance an application service provider (commonly referred to as an ASP) who then charges customers a rent for the access to the computer and software through some remote client system. These solutions have created some challenges for software publishers who have used the ‘per client’ licensing system. Since the licensing model favours more users within the same company or organization, the application service providers have been able to purchase just as many client licenses necessary in order for all their customers to run the applications they desire to run. But the customers of the application service providers are in many cases other companies and therefore the intention of the licensing model to supply discounted versions within a company could be considered broken. But is it considered broken legally just because the owner of the software rents his valid licenses to other companies? As hiring servers and software from application service providers has become a frequent way of outsourcing information technology departments and reducing costs for many companies most software publishers today incorporate legally valid paragraphs that handle these specific issues in their license agreements. And many have a specific license model that is to be used by the application service providers. So of course it is necessary for a licensing system as well to support these types of licenses.

2.10.3.12 GNU General Public License/Open source licenses

The GNU General Public License (GPL) is simply a licensing model designed for distribution of software where the source code and the software itself is available for anyone who desires it. The license assures anyone the right to change the source code and share the software with or without modifications to anyone. The software and the source code have to be free of charge, but a fee for covering the actual distribution costs or providing product warranty can be charged. The GNU GPL license states that the software is provided without any warranties unless otherwise is specified by the software publisher. If the GNU GPL licensed software or part of it is used as part of another software product, the GNU GPL license also applies to that software product. For further information about the GNU GPL licensing system see R.2.

2.11 ERP- and CRM-systems

ERP is short for Economic Resource Planning, so an ERP-system is typically an accounting system, a billing system, a system for managing budgets, a system for managing logistics and similar economically related tasks. It can be a system that handles only a few or several such tasks, integrated with other systems or just a stand-alone system.

CRM is short for Customer Relationship Management. A CRM-system includes any system where customer information can be stored and used for marketing purposes and/or the following up of new and/or existing customers. These systems typically keep track of documents, contact information and appointments with customers. CRM-systems are often integrated with ERP-systems, since both contain customer databases and it saves resources if such redundant information can be updated at the same time for both systems.

2.12 Summary

The KOPEK Payment System is based on an end user client that communicates data encrypted to a publicly available server, which will grant or deny access to content residing behind it, according to the end user's rights at that moment. Rights may or may not be purchased through the system, or administrators of the KOPEK server may manually assign rights. Payment for access rights is possible through common means for payment, but an account is also available in the KOPEK Payment System. The KOPEK account is an account specifically designed for the purpose of micro-payment where the user can deposit a small amount of money. With the KOPEK Payment System the user just has to trust that KOPEK AS doesn't abuse their payment information, since the companies they will be making their purchases from doesn't receive it. The KOPEK Payment System allows for easy revenue sharing with other registered KOPEK users whenever a customer makes a purchase.

To use the KOPEK Payment System for licensing of commercial software, some basic information regarding licensing and licensing models was defined in this chapter. Different requirements regarding copyright protection and possibilities for abuse of license models can still apply depending on the software publisher's policies. Some software publishers may require more information about each of their customers than what the KOPEK Payment System provides, because they require customer ownership. On the other hand, customers would often require as much privacy as possible, to avoid unwanted advertising, contact from the seller or abuse of their private information for marketing or other purposes.

3 Market Analysis / State Of The Art

3.1 Introduction

There are several solutions for license control on the market and only a selection of these are presented here. The selection is based on the tutors' input and the knowledge of the writer at the time of writing this report. Since there hasn't been any involvement of the publisher's of the solutions presented here, the information is only based on what has been published on the Internet or what could be deducted from a brief installation and review of the solutions.

3.2 Macrovision

The source of the information provided here was retrieved from the published information of Macrovision's web site (R.3) and the white paper "Electronic Licensing: The Build vs. Buy Decision" (R.4). Macrovision is one of the most experienced companies in the market for copy protection with more than 20 years of experience. Their products range through both hardware and software products for copy protection, not only for protecting computer software, but other digital products easily vulnerable to illegal distribution as well. In this report it is however only their software licensing systems that will be briefly examined. These products' primary focus is on preventing illegal distribution of the computer software while still preserving flexibility in choice of licensing models.

3.2.1 Macrovision FLEXnet Publisher

This is commercial software designed specifically for software licensing and distribution. Flexibility in choice of use and complexity is available through a variety of modules that can be combined for almost any purpose within software licensing, distribution and copy protection. The licensing module offers the ability to deploy software with three different pricing models (for "heavy", "medium" and "light" users) in more than 1000 different licensing models. The licensing models mentioned in the datasheet provided by Macrovision (R.5) are:

"Locked" – This is the corresponding license model to the "single computer license" described in chapter 2.10.3.4 where the license is "locked" to the hardware on which it will be run.

"Floating" – Similar to the "per client licensing model" from chapter 2.10.3.6, a limited number of licenses can be used simultaneously.

"Named User" – Corresponds to the "personal license" detailed in chapter 2.10.3.3 where each person that is to use the software has one license.

"Site" – Since the details of what this license model includes are not described in the datasheet from Macrovision, an assumption can only be made that this is a license model limiting the use of the software to an enterprise/company or a department of an enterprise/company. It could also be a limitation to run the software on certain network addresses. Either way, it probably is a variant of the site license model outlined in chapter 2.10.3.9.

“Subscription” – A rented license or a time limited license that is renewed by payment after expiration, similar to the license model in chapter 2.10.3.9.

“Time limited” – The same license model as described in chapter 2.10.3.1.

In addition, the Macrovision FLEXnet Publisher licensing module offers the possibility of offering temporarily detached use of a network licensed product, useful for laptop computer users that sometimes operate without the possibility to contact a license server.

Macrovision claims that change of licensing model can be done without software modification and that pricing and terms are easily altered. The products can be sold as “lite”, “standard” or “premium” versions where different features are available accordingly. Whether it is possible to have additional versions than those three so that this could be used for selling module based software licenses of more than three modules could not be extracted from the Macrovision product datasheet. But the Macrovision FLEXnet Publisher can be implemented as a license control system in the source code or as a wrapper around the software it is to protect. In the case of source code implementation module based licensing could at least theoretically be supported.

The Macrovision solution is – as far as one can extract from the published information on their web site – primarily based on licensing server software that all end users of the product using Macrovision FLEXnet Publisher for license control has to have. There doesn’t seem to be any central Internet based license or product database, but as the license server could be available on the Internet as well it would be possible to have any licenses available wherever Internet is available. The solution could also be implemented with the use of hardware keys for even stronger copy protection, but this is part of another module of the software called “Enhanced security module”, which is sold separately.

Any technical documentation on how payment of licenses and retrieving additional licences is done could not be found on the Macrovision web site.

3.3 SafeNet

SafeNet is another well-known provider of software protection systems, mainly based on hardware keys. SafeNet has been in the market for software protection since 1984. The background information for this brief overview of their solution was found on the SafeNet web site (R.6).

3.3.1 Sentinel RMS

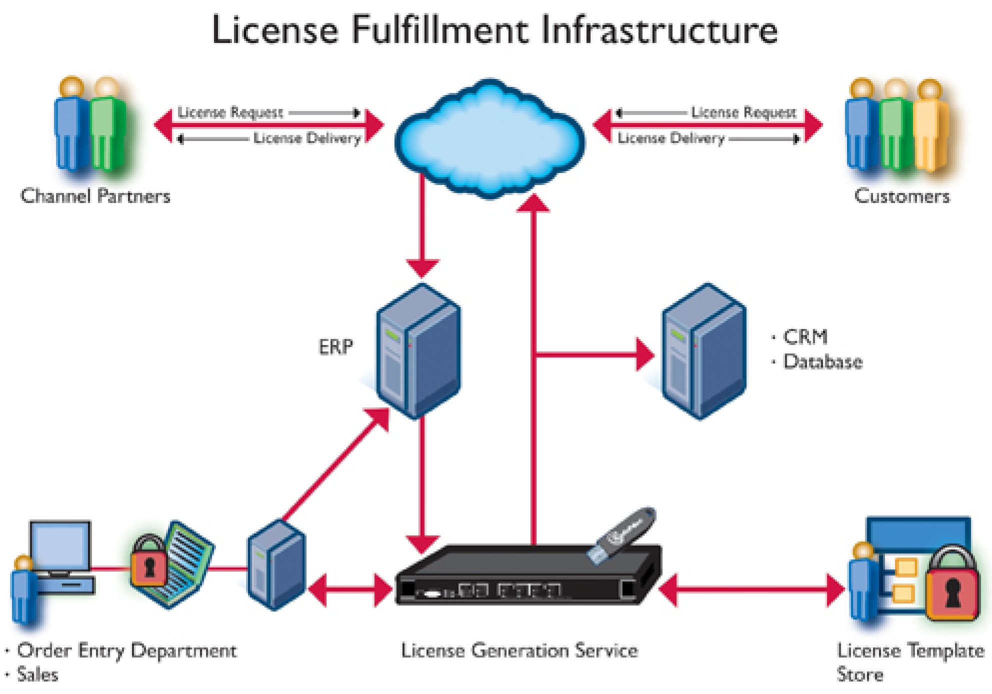
The Sentinel RMS (Rights Management System) is based on either source code implementation of the license control system or a shell that is wrapped around an existing application. It supports several different license models like evaluation, feature based, site, pay-per-use and rented licenses.

According to the SafeNet product information presentation, the Sentinel RMS requires an activation of licenses after installation of the software that is under the systems control. It also requires a license server installation, which can be installed quickly and without any need for configuration. The advantage is that this gives the end user the opportunity to see and manage own licenses.

The system allows something the SafeNet product information presentation calls “grace period licensing”, which is a period of time the legitimate users can continue to use licenses in order to avoid disruption in case of license failure.

Integration with ERP and CRM systems is possible, but has to be customized for each system. This could be useful for automation of order handling and updating of licenses, but also for the distribution of the information gathered by the license control system to a broader range in the organization than the part directly involved with license handling. Licensing information can also be stored centrally, which will further improve the software publisher’s control of the customers’ licenses.

Figure 3.1: Sentinel RMS overview (source of figure is SafeNet web site, R.6)



3.4 SWREG

SWREG is a simpler software license distribution organization than Macrovision and SafeNet. The system provides no direct copy protection, but an easy way of selling and distributing software files, license files or license keys to customers. The information has been found on the SWREG web site (R.7).

This solution allows a software publisher to either upload the software files in a package to the SWREG server or upload a single license file to the SWREG server. It is also possible to just upload valid licenses key codes to the SWREG server, or have both software or license files and valid license keys uploaded to the SWREG server. The SWREG server will then make sure the end users are charged before they can either download the software package file, the license file and/or the valid license key. SWREG will eventually pay the software publisher's their profit after the SWREG fees and profit has been calculated and charged.

All sales are immediately reported by SWREG to the software publisher by email, so some level of control of issued licenses can be maintained.

The solution is simple, but since it is easy to implement and easy for end users to understand, this has become a popular way of selling low cost software applications. It can of course be combined with other licensing control systems in order to improve the copy protection.

3.5 Summary

Macrovision and SafeNet seem to offer licensing solutions that would be satisfactory for most larger software publishers. Their solutions focus on protecting the software against illegal distribution or copy protection. There also exists many other providers of similar solutions, but none of these has been as successful in terms of market position as Macrovision and SafeNet, so they are not included in this report.

Still, there doesn't seem to be an optimal solution towards complete electronic sales and distribution of software, there is no way of adjusting or setting prices and discounts for individual sales or sharing revenue with resellers or distributors automatically.

It seems obvious that Macrovision and SafeNet customers mainly have traditional ways of selling the software, and that payment handling is done in an external system. SWREG on the other hand seems to be missing all aspects related to copy protection directly in their solution, but the solution handles electronic sales. Sharing revenue with distributors or resellers must be done manually. The SWREG solutions seem suitable for software publishers that supply product with no need for individual negotiation of prices and terms for each customer. In addition, finding a good solution for copy protection is left to the software publisher.

Summarized, it could seem that there is a void in the market for solutions supplying both copy protection and the possibility of electronic sales and distribution.

4 Using the KOPEK Payment System As Is

4.1 Introduction

Using the KOPEK Payment System as a licensing solution for software just based on the API of the client software and the KOPEK Payment Server is possible without any modifications to the KOPEK components. But of course the software has to be implemented or modified so the necessary lines of code that performs the license check are included. This chapter will give a brief walk-through of what code is required and what kind of license control this can achieve. Since the KOPEK system is based on an Internet connection to the server, the code has to include some functions for communicating with the KOPEK Payment Server.

4.2 *Overview of the KOPEK Payment System*

The following schematics illustrate how the KOPEK Payment System works and how it can be implemented.

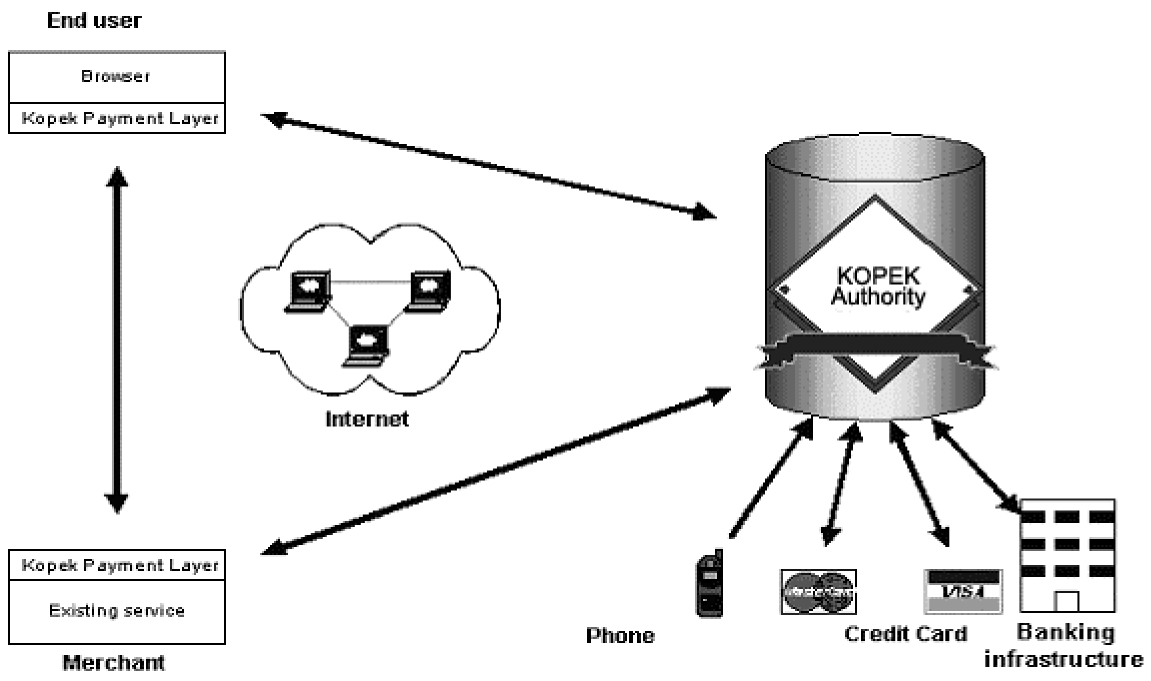
4.2.1 Overview of the KOPEK Payment System basic design

The KOPEK Payment System works in the following way:

- A centralized KOPEK Authority server contains all registered KOPEK users and keeps track of their accounts and balances. This is also the server that authenticates users when they log on from an end user client. The KOPEK Authority also communicates with the payment institutions that are available for use in the KOPEK Payment System.
- The KOPEK client communicates with both the KOPEK Authority as well as any KOPEK Payment Server installed at a merchant that is using the KOPEK Payment System.
- Whenever a KOPEK user tries to access content protected by a KOPEK Payment Server, the KOPEK Payment Server also has to communicate with the KOPEK Authority.

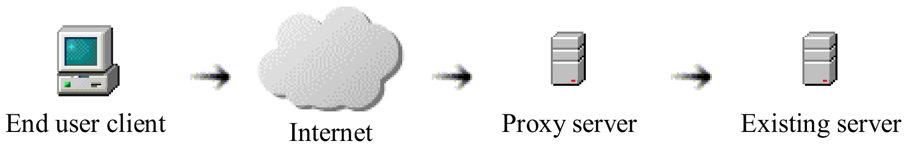
Figure 4.1 shows the communication between the different components of the KOPEK Payment System.

Figure 4.1: The KOPEK Payment System overview (figure from the KOPEK Payment Server documentation)



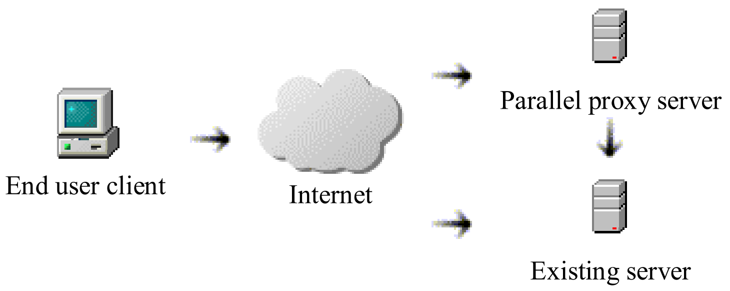
4.2.2 KOPEK Payment Server configurations

Figure 4.2: Proxy-type server (figure from the KOPEK Payment Server documentation)



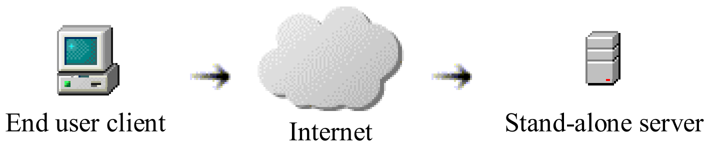
In this configuration, an existing server contains the content that is to be sold through the KOPEK Payment System. The KOPEK Payment Server is installed in front of the existing server so all traffic to and from the existing server has to go through it. That makes it possible for the KOPEK Payment Server to filter the traffic and hence make sure any content that shouldn't be allowed to the end user client never reaches the end user client.

Figure 4.3: Parallel proxy-type server (figure from the KOPEK Payment Server documentation)



This configuration allows an implementation of the KOPEK Payment System without interfering with any existing solutions. The content on the existing server can be accessed either directly from the existing server or through the KOPEK Payment Server. However, the KOPEK Payment Server will not be able to stop content on the existing server that shouldn't reach an end user client to reach it if the end user client accesses the existing server directly. So the existing server has to have its own access control system/protection system if this content is to be protected.

Figure 4.4: Stand-alone server (figure from the KOPEK Payment Server documentation)



In the stand-alone server configuration, the KOPEK Payment Server is used as a web server as well as an access control/protection system for the content that resides on it. The KOPEK Payment Server only support limited web server functionality, so this is not a solution meant for advanced web content.

4.2.3 Overview of the KOPEK Payment System from an end user's perspective

What really happens when the KOPEK Payment System is implemented and a client tries to retrieve content on it or behind it is the following:

- A request for the content is sent from the end user client for the desired content. The request has to go through the KOPEK Payment Server since the content is located behind it or on it.

- The KOPEK Payment Server will check if the request is for content that it protects or for content that is publicly available. If the content is publicly available, the end user client will receive the requested content as if it came from any other server.

- If the KOPEK Payment Server protects the content requested, the KOPEK Payment Server returns a request to the end user client for the users identification in the KOPEK Payment Client. If the KOPEK Payment Client isn't already running, the end user has to start it and log on to it (figure 4.5). In most cases the end user's web browser will start the client automatically. In the scenario where the KOPEK Payment Client has never been installed on the end user's computer the end user will be prompted to download and install it before proceeding. The first time a new end user will log on to the KOPEK Payment System, a simple registration of identification credentials has to be made (figures 4.6-4.8). That is unless the end user has an electronic identification token issued by one of the trusted authorities the KOPEK Payment System supports (see chapter 2.4 *"Identification certainty by authentication"*).

Figure 4.5: The KOPEK Payment Client end user authentication box

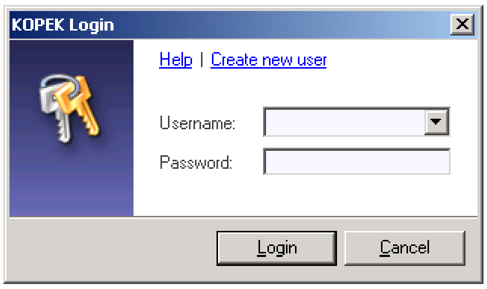


Figure 4.6: The KOPEK Payment Client new end user registration form

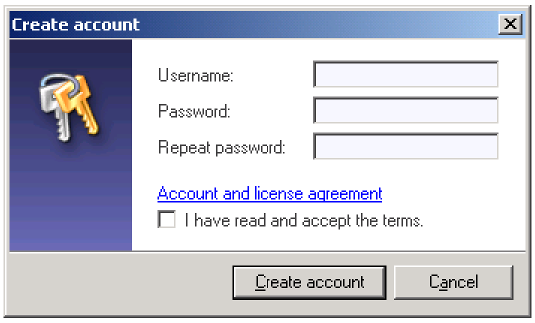
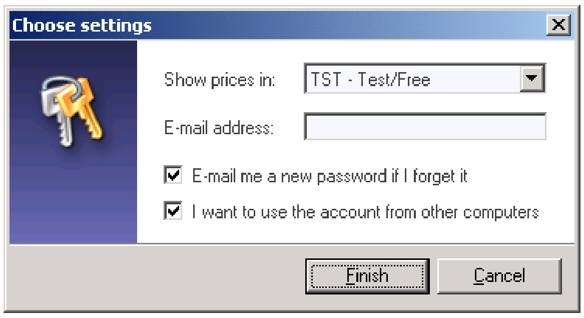


Figure 4.7: KOPEK Payment Client second registration form when creating a new end user



Choose settings

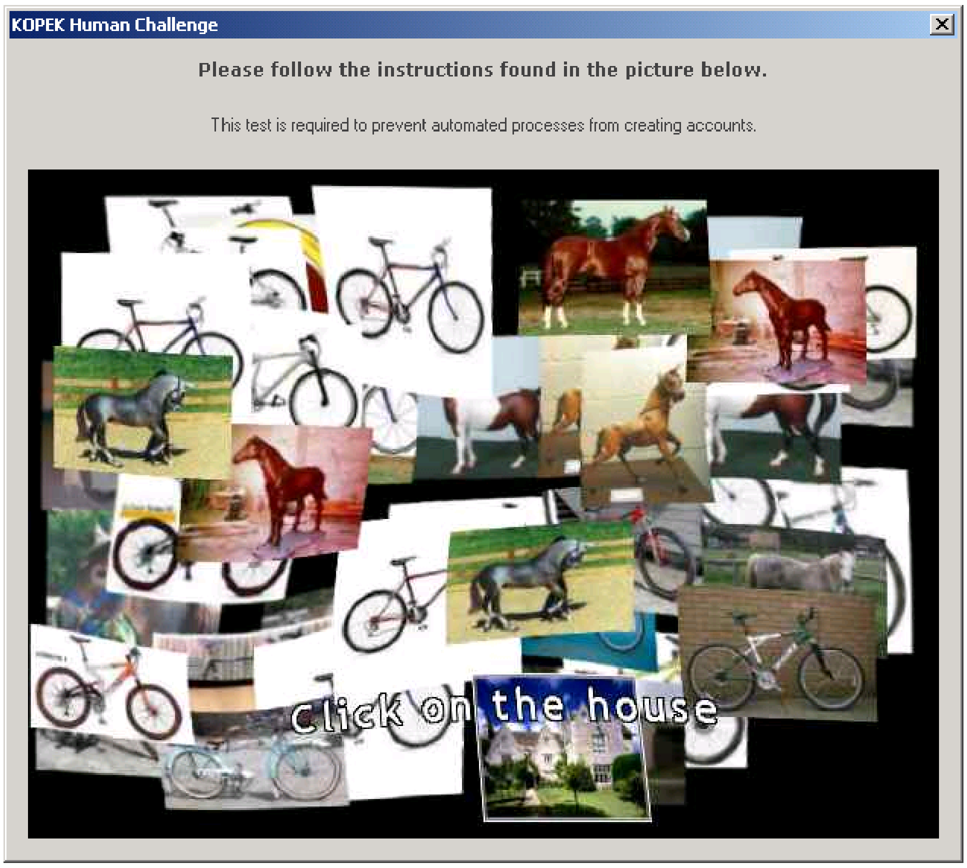
Show prices in: TST - Test/Free

E-mail address:

☑ E-mail me a new password if I forget it

☑ I want to use the account from other computers

Figure 4.8: KOPEK Payment Client final step of new end user registration



KOPEK Human Challenge

Please follow the instructions found in the picture below.

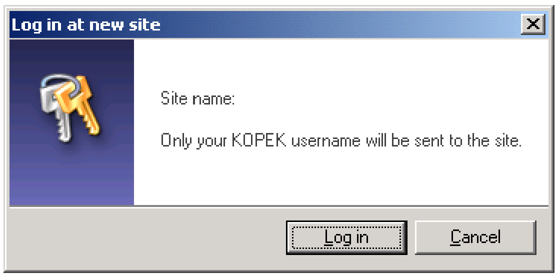
This test is required to prevent automated processes from creating accounts.

To make sure that it is a human being that is creating the KOPEK user account and not some automated machine process, the user is asked to identify a randomly selected object amongst several other objects by clicking on it.

- The authenticated user is returned and the KOPEK Payment System checks whether or not the end user should be granted access to the requested content. If the access is granted, the result is returned as if no access control system was present. However if access is not granted, the KOPEK Payment System returns the options available to the end user in order to get access if any (figure 4.10-4.11). If the end user doesn't comply with any of these options or there are no options available to that end user, a message that explains that access has been denied will be displayed to the end user (figure 4.12).

When access to a site is granted provided only that the KOPEK username can be submitted and the site has not previously been accessed using the KOPEK Payment System an information dialog appears (figure 4.9).

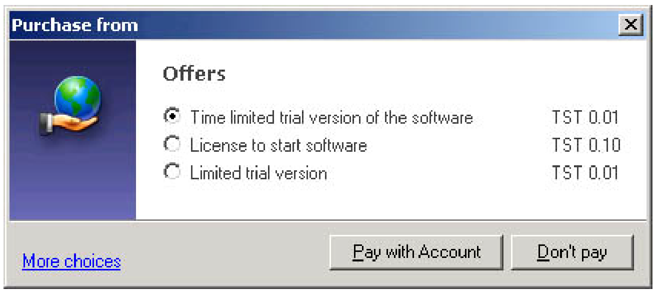
Figure 4.9: KOPEK Payment Client new site information window



Site name:

Only your KOPEK username will be sent to the site.

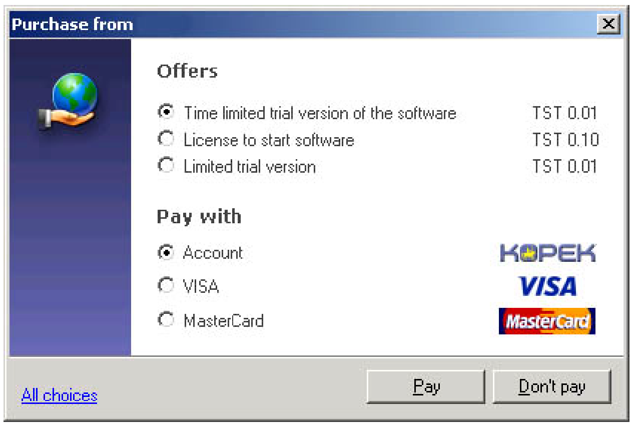
Figure 4.10: Simple options available to an end user when access isn't directly granted



Offers

[✓ Time limited trial version of the software]	TST 0.01
License to start software	TST 0.10
Limited trial version	TST 0.01

Figure 4.11: Advanced options available to an end user when access isn't directly granted



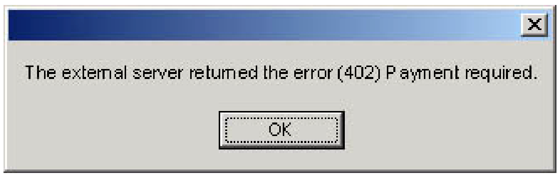
Offers

[✓ Time limited trial version of the software]	TST 0.01
License to start software	TST 0.10
Limited trial version	TST 0.01

Pay with

[✓ Account]
VISA
MasterCard

Figure 4.12: Error message returned when the end user doesn't comply with any of the options

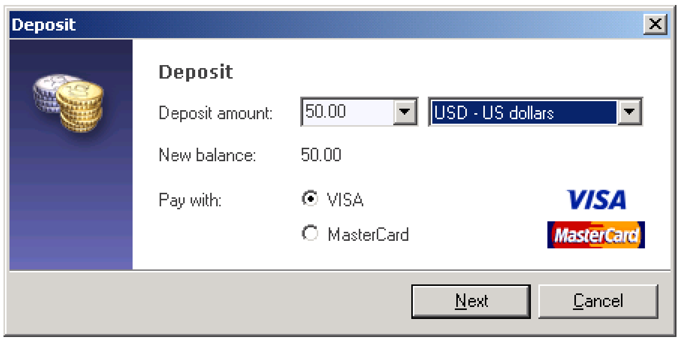


The external server returned the error (402) Payment required.

4.2.4 The KOPEK Payment Client account operations

The following figures (4.13-4.16) shows the dialogs that can be accessed from the KOPEK Payment Client in order to deposit money from a VISA card to the KOPEK account that is available to all KOPEK users. Whether or not an end user wants to deposit and use the KOPEK account at all is optional, it is possible to simply charge a credit card or a cell phone directly each time a purchase is made.

Figure 4.13: KOPEK Payment Client deposit money dialog


Deposit

Deposit

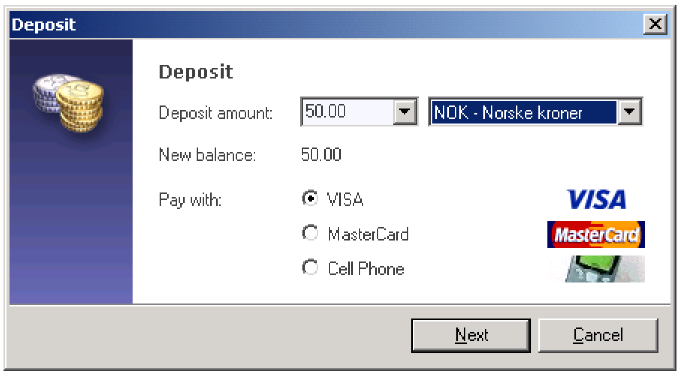
Deposit amount: 50.00 USD - US dollars

New balance: 50.00

Pay with: (●) VISA ( ) MasterCard

Depositing US dollars is only supported by credit card in the KOPEK Payment Client.

Figure 4.14: KOPEK Payment Client Deposit money dialog using Norwegian currency


Deposit

Deposit

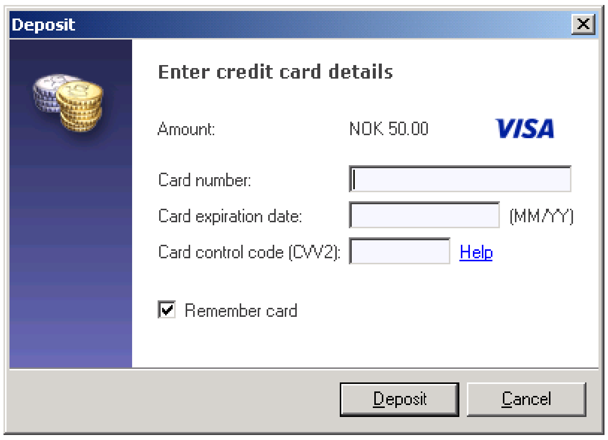
Deposit amount: 50.00 NOK - Norske kroner

New balance: 50.00

Pay with: (●) VISA ( ) MasterCard ( ) Cell Phone

When depositing Norwegian currency, it is also possible to deposit from a cell phone.

Figure 4.15: KOPEK Payment Client deposit money from VISA card dialog


Deposit

Enter credit card details

Amount: NOK 50.00

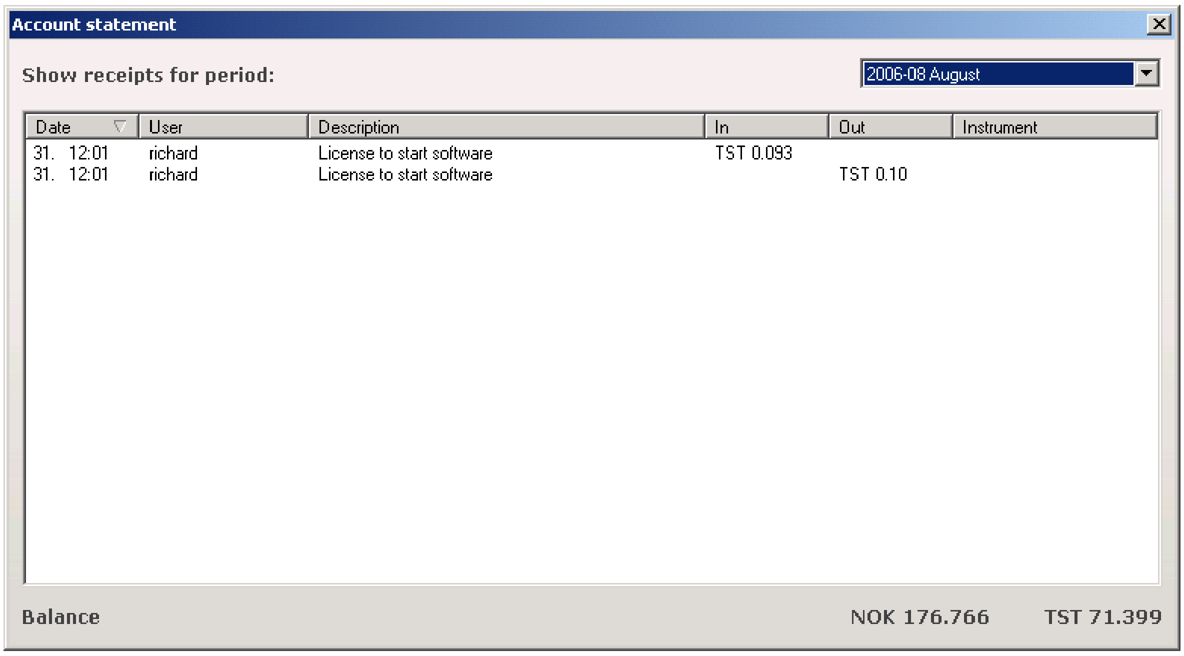
Card number:

Card expiration date: (MM/YY)

Card control code (CVV2): Help

[✓] Remember card

Figure 4.16: KOPEK Payment Client account overview


Account statement

Show receipts for period: 2006-08 August

Date	User	Description	In	Out	Instrument
31. 12:01	richard	License to start software	TST 0.093		
31. 12:01	richard	License to start software		TST 0.10	

Balance NOK 176.766 TST 71.399

The account statement will show all transactions made by using the KOPEK Payment System, transactions debited directly from a credit card, a cell phone or from the KOPEK account.

4.3 Internet communication class

The following C# class contains the required code for communicating with the KOPEK Payment Server if a C# application is to use the KOPEK Payment System to verify that the end user of the application has a valid license:

```
using System;
using System.Security.Cryptography;
using System.Web;
using System.Net;
using System.Text;
using System.IO;
using System.Threading;

namespace TheNameSpaceNameWanted
{
    // Change these values to suit your setup.
    class Configuration
    {
        public static string GetContentServerHost ()   { return "host.domain.name"; }
        public static string GetLicenseControlScript() { return "/getlicense.php"; }
    }

    /// <summary>
    /// Access Check object. Checks access and notifies parent.
    /// </summary>
    public class KOPEKFunctions
    {
        // Generate your own 1024-bit RSA key:
        // openssl genrsa -out key.pem 1024 (requires OpenSSL from www.openssl.org)

        // Output modulus:
        // openssl rsa -in key.pem -modulus -noout
        // Use a keyboard macro or script to convert to the below format:
        static private byte[] s_pubKeyModulus =
        {
0xC6,0xCB,0x21,0x24,0x7D,0xAB,0xDD,0x99,0x4B,0xE1,0xF2,0xAA,0x38,0x53,0x87,0x72,0x2A,
0x25,0x00,0xEF,0xCD,0xEF,0x18,0x48,0xD2,0x80,0x97,0xF1,0xFB,0xD6,0x29,0x86,0xC1,0xA6,
0x6D,0xAD,0x6F,0x27,0xE6,0xEC,0xC7,0xE1,0x11,0x7E,0x71,0x19,0x09,0xDF,0xAE,0x2D,0xC7,
0x6F,0x72,0x7C,0x1A,0x19,0x19,0x65,0x83,0x39,0x3C,0x68,0xA6,0x23,0x0F,0x94,0xE5,0x8B,
0x14,0x25,0xE8,0x76,0x12,0x3B,0x91,0xCF,0x27,0x2E,0xC4,0x7B,0x57,0x7B,0x6B,0x04,0xEB,
0x41,0x82,0xF6,0xB5,0x7D,0xD0,0xCD,0xD7,0x9D,0xE4,0x9A,0x9A,0x11,0x4C,0xF0,0x0D,0x5F,
0x36,0xC3,0xC9,0xCF,0xCB,0xF1,0x63,0x07,0xCB,0xA7,0xBD,0x3E,0x8A,0x4A,0x35,0xCF,0x8C,
0x19,0x1D,0xF3,0x88,0xA9,0x12,0x89,0x15,0x81
        };

        // Default openssl RSA exponent 65537 (0x10001)
        static private byte[] s_pubKeyExponent = { 0x01, 0x00, 0x01 };

        private RNGCryptoServiceProvider m_random;
        private RSACryptoServiceProvider m_rsa;
        public int loggedOnKOPEKUserID;


        public KOPEKFunctions()
        {
            // Setup random source
            m_random = new RNGCryptoServiceProvider();

            // Setup RSA public key
            m_rsa = new RSACryptoServiceProvider();
            RSAParameters keyInfo = new RSAParameters();
            keyInfo.Modulus  = s_pubKeyModulus;
            keyInfo.Exponent = s_pubKeyExponent;

            m_rsa.ImportParameters(keyInfo);
        }

        /// <summary>
        /// Verify that signature was created using key file used to
        /// generate public key in this code
        /// </summary>
        /// <param name="dataStr">Challenge sent to server side script</param>
        /// <param name="base64Signature">Response signature from server side script</param>
        /// <returns>Match between challenge and signature</returns>
        public bool CheckSignature (string dataStr, string base64Signature)
        {
            byte[] signature = Convert.FromBase64String(base64Signature);

            UTF8Encoding ue = new UTF8Encoding();
            byte[] data = ue.GetBytes(dataStr);

            SHA1 sha = new SHA1CryptoServiceProvider();
            byte[] hash = sha.ComputeHash(data);

            RSAPKCS1SignatureDeformatter deformatter = new RSAPKCS1SignatureDeformatter(m_rsa);
            deformatter.SetHashAlgorithm("SHA1");

            return deformatter.VerifySignature(hash, signature);
        }
```

```
        /// <summary>
        /// Perform HTTP request through the KOPEK payment system.
        /// This yields access control and payment if needed.
        /// </summary>
        /// <param name="uri">Path to server side license check script</param>
        /// <returns>Answer from server side script</returns>
        public string OnlineAccessCheck (string uri)
        {
            string host = Configuration.GetContentServerHost();
            string signature = null;
            string sigPrefix = "Signature: ";
            string clientPrefix = "http://localhost:27504/KSecure____";
            string url = clientPrefix + "http://" + host + uri;

            HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);
            try
            {
                request.Timeout = 9000 * 1000; // 9000 seconds (we want ~unlimited)

                HttpWebResponse response = (HttpWebResponse)request.GetResponse();
                StreamReader sr = new StreamReader(response.GetResponseStream());
                for(;;)
                {
                    string line = sr.ReadLine();
                    if(line == null)
                       break;

                    if(line.StartsWith(sigPrefix))
                       signature = line.Substring(sigPrefix.Length);
                }

                sr.Close();
                response.Close();
            }
            catch(ThreadAbortException)
            {
                request.Abort();
            }

            return signature;
        }

        /// <summary>
        /// We append a random challenge string to each request to make it
        /// impossible to defeat the access control through replay attacks.
        /// </summary>
        /// <returns>Random challenge string</returns>
        public string GetChallengeString ()
        {
            byte[] randomBytes = new Byte[30];
            m_random.GetBytes(randomBytes);

            return Convert.ToBase64String(randomBytes);
        }

        /// <summary>
        /// Check access using configuration settings
        /// </summary>
        /// <returns>Result of access check</returns>
        public bool CheckAccess(string productID)
        {
            string challenge = GetChallengeString();

            string uri = Configuration.GetLicenseControlScript() +
                "?productid=" + HttpUtility.UrlEncode(productID) +
                "&challenge=" + HttpUtility.UrlEncode(challenge);
            string signature = OnlineAccessCheck(uri);
            bool accepted = CheckSignature(HttpUtility.UrlEncode(challenge), signature);

            if(!accepted)
            {
            }
            return accepted;
        }
    }
}
```

The class ensures a secure communication with the server all the way to the web server where the KOPEK Payment Server resides. On the web server the following script must reside (in this case a PHP script is used):

```php
<?php

header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
header("Cache-Control: no-store, no-cache, must-revalidate");
header("Cache-Control: post-check=0, pre-check=0", false);
header("Content-type: text/html");
header("Pragma: No-cache");

assert(get_magic_quotes_gpc());

$challenge = $_GET['challenge'];

if((strlen($challenge) < 5) || (strlen($challenge) > 512))
{
    echo "Error: Invalid challenge";
    exit;
}

if(strlen($challenge) > 0)
{
   $tempFile = tempnam("/tmp", "challenge");

   $handle = fopen($tempFile, "w");
   fwrite($handle, substr($_SERVER['REQUEST_URI'], strpos($_SERVER['REQUEST_URI'],
"challenge=") + 10)
);
   fclose($handle);
   echo "Signature: " . exec("cat \"" . $tempFile  . "\" | openssl sha1 -sign licensekey.pem |
openssl base64 -e | awk '{ printf $0 }'") . "\r\n";
   unlink($tempFile);
}
?>
```

The name of the script must correspond to the name returned by the GetLicenseControlScript() function of the Configuration class in the C# code. So in this case the name of the script is getlicense.php. The only function of the script is to use the private encryption key to generate a signature so that it matches the challenge string sent from the C# application to the script as a parameter. This is what ensures that the C# application is talking to the correct web server and not some attempt to fake a KOPEK Payment Server with another web server. The private part of the key file (here named licensekey.pem) also has to reside on the server and the public part of the key file must be incorporated in the C# code as the byte array named s_pubKeyModulus.

4.4 How the KOPEK Payment System works

4.4.1 Internet web server request

When the application calls the function CheckAccess("ProductID") of the C# code given in the previous section, the application will trigger an attempt to contact the web server returned by the GetContentServerHost() function of the Configuration class to get the response returned by the script with the path and name returned by the GetLicenseControlScript() function. The script is called with at least one parameter, the challenge string from which it will generate a signature to return. But in this case another parameter is given, the parameter for identifying which product the application is checking for valid license for. This makes it possible for the same application to use the same server script for different parts of the application if they require different licenses to be valid.

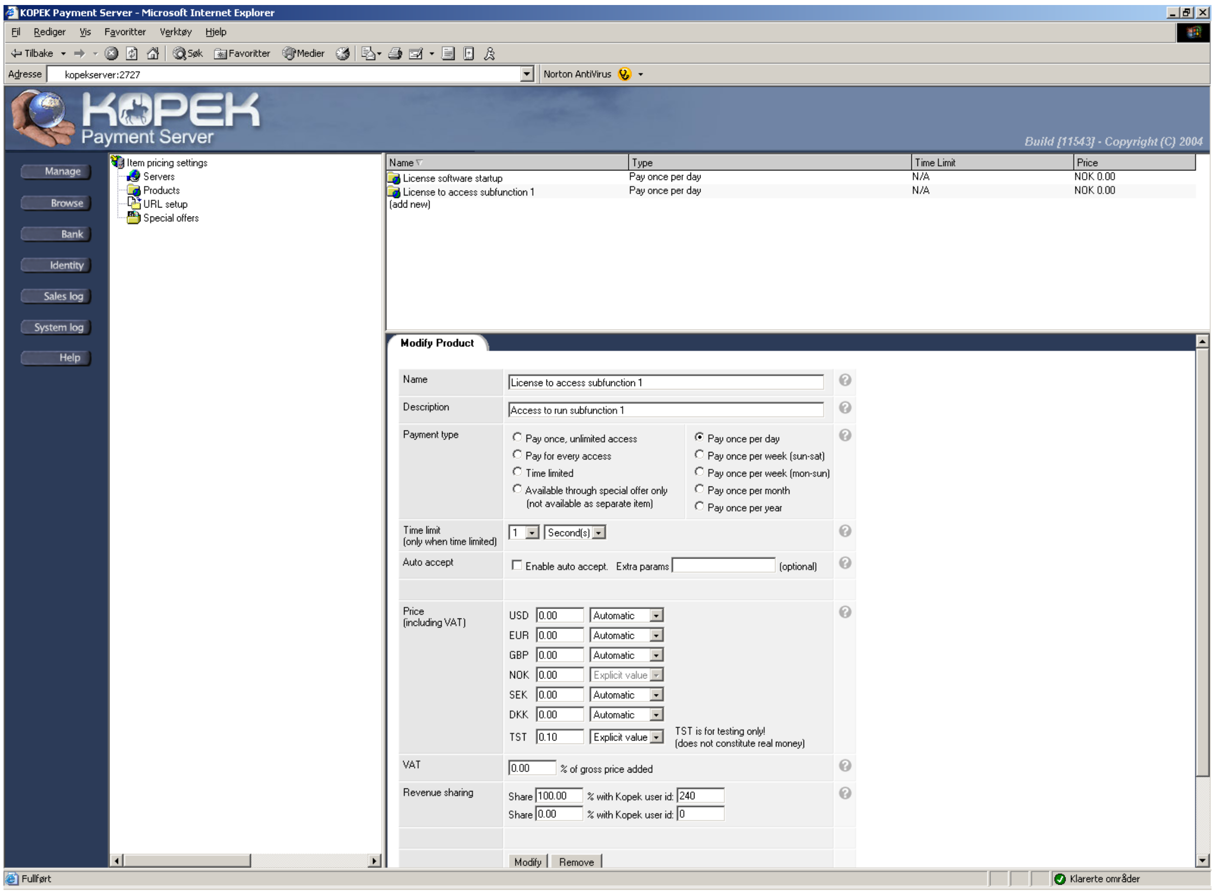
4.4.2 License validation procedure

But how does the application verify that the license is valid just based on the response signature returned from the server script? The answer is simple but requires an understanding of the basics of the KOPEK Payment System. If the license isn't valid, the KOPEK Payment Server (which the web request from the application has to pass through) won't find that the KOPEK user logged on at the computer where the application is run can be granted access to run the script at all. It won't even pass on the request to the web server where the script actually resides. It will simply trigger the KOPEK client software, which is running on the computer where the user is trying to launch the application or part of the application that requires a valid license, to display a dialog for the user. This dialog will prompt the user for payment if that is what is required in order to retrieve a valid license. If the user accepts the request for payment, the payment is validated first and then the web request is passed on to the web server where the server script resides. So the correct signature is returned and the application can verify this. If the user doesn't accept the payment request or the payment validation fails, the KOPEK Payment Server returns a web response with the reason why the request cannot be passed on to the script. The application of course searches the response for the correct signature because it doesn't know that the response doesn't come from the server script. Since the signature can't be found, the application knows that there isn't a valid license for this user and the user is denied access to proceed. Whether the application will pass on the message with the reason from the KOPEK Payment Server to the user or not is of course up to the application developer to determine. The application can return its own responses if desired by the publisher.

4.4.3 KOPEK Payment Server configuration

Knowing how the KOPEK Payment System verifies the license still leaves the question of how the KOPEK Payment Server can be configured in order to come up with the correct response. This requires a deeper understanding of the KOPEK Payment Server. In the '*Theory and background'* chapter, the sub chapter '*KOPEK Payment System*', a brief introduction was given in the paragraph '*KOPEK Payment Server*'. In practice, what has to be done after the installation of the KOPEK Payment Server software is to define the different products, that is, the name and price of each product item that will be available from the system. As figure 4.17 shows, there are several possibilities for configuring different models of pricing and what the price should be in different currencies. An explicit value can be used for each currency or the currency price can be calculated automatically based on one explicit value. In addition one or more of the available currencies can be disabled if that is desired. It is also possible to define products that are almost free of charge by simply setting the price to for instance 0.01.

Figure 4.17: Configuring KOPEK Payment Server products



Adresse: kopekserver:2727

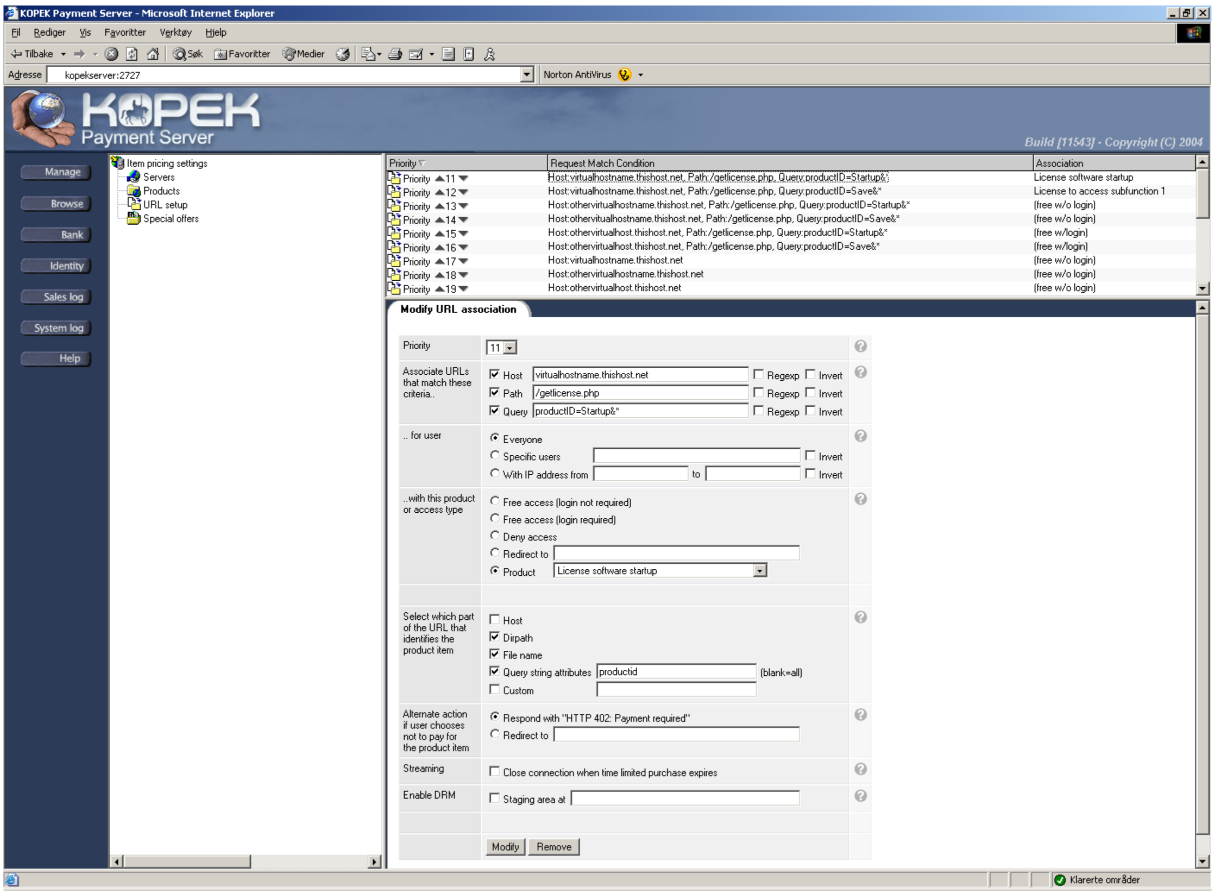
Name	Type	Time Limit	Price
License software startup	Pay once per day	N/A	NOK 0.00
License to access subfunction 1	Pay once per day	N/A	NOK 0.00
(add new)			

Modify Product

Name	License to access subfunction 1
Description	Access to run subfunction 1
Payment type	( ) Pay once, unlimited access ( ) Pay for every access ( ) Time limited ( ) Available through special offer only (not available as separate item) [✓ Pay once per day] ( ) Pay once per week (sun-sat) ( ) Pay once per week (mon-sun) ( ) Pay once per month ( ) Pay once per year
Time limit (only when time limited)	1 Second(s)
Auto accept	☐ Enable auto accept. Extra params (optional)
Price (including VAT)	USD 0.00 Automatic EUR 0.00 Automatic GBP 0.00 Automatic NOK 0.00 Explicit value SEK 0.00 Automatic DKK 0.00 Automatic TST 0.10 Explicit value — TST is for testing only! (does not constitute real money)
VAT	0.00 % of gross price added
Revenue sharing	Share 100.00 % with Kopek user id: 240 Share 0.00 % with Kopek user id: 0
	Modify Remove

This explains how the KOPEK Payment Server knows what product name and price it can display. But the connection between the web request and the products hasn't been established yet. This is done by defining which URL will represent which product (see figure 4.18). When one user pays for one product the KOPEK Payment Server remembers this so the user isn't prompted to pay again until the period for which the payment made expires. If there is no time limitation on the product, the user will never be charged again for that product. And of course, if the product pricing policy is changed from having no time limitation to a time limitation or if the time limit period is reduced, the policy that was active during the time of the sale to the existing customers stand until their respective time limitations expire. There is however some ways to override this functionality. If the URL filters are modified so that the product is identified in another way than it originally was, the customer may be charged as if his original purchase was never made. The same thing happens if the product definition is removed from the KOPEK Payment Server. Changing the URL filter back again will however make the original purchase valid once again, but adding a deleted product definition once more will not. For a software license provider this can be considered a weakness, since the software may want to check the payment server for a valid payment quite often. And customers probably will be very insulted if they are asked to pay more often than promised at the time of purchase or renewal of the purchase.

Figure 4.18: Configuring KOPEK Payment Server filters



Priority	Request Match Condition	Association
Priority 11	Host:virtualhostname.thishost.net, Path:/getlicense.php, Query:productID=Startup&*	License software startup
Priority 12	Host:virtualhostname.thishost.net, Path:/getlicense.php, Query:productID=Save&*	License to access subfunction 1
Priority 13	Host:othervirtualhostname.thishost.net, Path:/getlicense.php, Query:productID=Startup&*	(free w/o login)
Priority 14	Host:othervitualhostname.thishost.net, Path:/getlicense.php, Query:productID=Save&*	(free w/o login)
Priority 15	Host:othervirtualhost.thishost.net, Path:/getlicense.php, Query:productID=Startup&*	(free w/login)
Priority 16	Host:othervirtualhost.thishost.net, Path:/getlicense.php, Query:productID=Save&*	(free w/login)
Priority 17	Host:virtualhostname.thishost.net	(free w/o login)
Priority 18	Host:othervirtualhostname.thishost.net	(free w/o login)
Priority 19	Host:othervirtualhost.thishost.net	(free w/o login)

Modify URL association

Priority	11
Associate URLs that match these criteria..	☑ Host virtualhostname.thishost.net ☐ Regexp ☐ Invert ☑ Path /getlicense.php ☐ Regexp ☐ Invert ☑ Query productID=Startup&* ☐ Regexp ☐ Invert
.. for user	[✓ Everyone] Specific users ☐ Invert With IP address from to ☐ Invert
..with this product or access type	Free access (login not required) Free access (login required) Deny access Redirect to [✓ Product] License software startup
Select which part of the URL that identifies the product item	☐ Host ☑ Dirpath ☑ File name ☑ Query string attributes productid (blank=all) ☐ Custom
Alternate action if user chooses not to pay for the product item	[✓ Respond with "HTTP 402: Payment required"] Redirect to
Streaming	☐ Close connection when time limited purchase expires
Enable DRM	☐ Staging area at
	Modify Remove

4.4.4 Handling different licensing models

This chapter examines the details of how the different licensing models outlined in chapter 2.10.3 can be handled by the KOPEK Payment System just by using the system the way it is prior to any modifications that could adapt it further for the use as a licensing system. Both strengths and weaknesses will be considered.

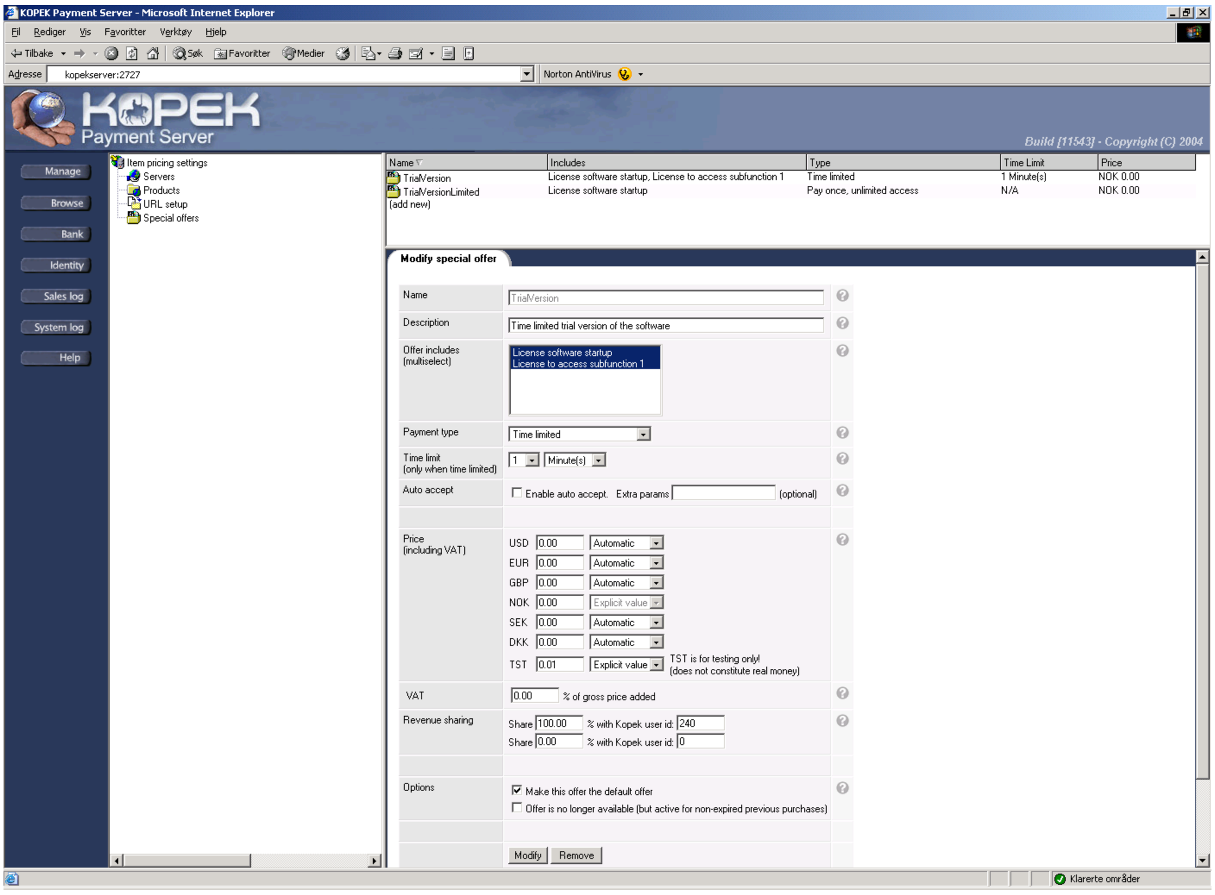
4.4.4.1 Time-limited license

In chapter 2.10.3.1 the concept of a time-limited license model was described. The KOPEK Payment System supports time-limited licenses if desired, by supporting different levels of identification security in combination with a custom time limitation on a given product. The levels of identification include two of the recently developed standards for electronic signatures in Norway, BankID and BuyPass. KOPEK has even been a partner in the BankID project, which mainly has been developed by Norwegian banks. The BankID standard is built on the PKI (Public Key Infrastructure) technology (further explained in chapters 2.5-2.6). A time-limited license of desired security can be implemented by applying the appropriate level of identification security. In addition checking the validity of the license with the KOPEK Payment Server at strategic points of time further enhances security. Issues regarding security of time-limited licensing are discussed in chapter 2.10.3.2.

4.4.4.2 Demo/trial license

This type of license is probably quite self-explicable, but refer to chapter 2.10.3.2 for more details on what a demo/trial license model is. Handling demo/trial licenses with restriction in software functionality is straight forward with the KOPEK Payment System. By building the software as different modules which each require its own purchased license, a special offer can be made for the selected modules that should be included in a demo/trial version of the software (see figure 4.19). This also enables the software publisher to very easily reconfigure which modules should be included in the demo/trial license version of the software. And different configurations of modules can of course be offered at the same time for different potential customers if that is desired. If the software isn't suited for modules, time limitation will of course be the alternative to consider first.

Figure 4.19: Configuring KOPEK Payment Server special offers



Name	Includes	Type	Time Limit	Price
TrialVersion	License software startup, License to access subfunction 1	Time limited	1 Minute(s)	NOK 0.00
TrialVersionLimited	License software startup	Pay once, unlimited access	N/A	NOK 0.00
(add new)				

Modify special offer

Name	TrialVersion
Description	Time limited trial version of the software
Offer includes (multiselect)	License software startup License to access subfunction 1
Payment type	Time limited
Time limit (only when time limited)	1 Minute(s)
Auto accept	☐ Enable auto accept. Extra params (optional)
Price (including VAT)	USD 0.00 Automatic EUR 0.00 Automatic GBP 0.00 Automatic NOK 0.00 Explicit value SEK 0.00 Automatic DKK 0.00 Automatic TST 0.01 Explicit value TST is for testing only! (does not constitute real money)
VAT	0.00 % of gross price added
Revenue sharing	Share 100.00 % with Kopek user id: 240 Share 0.00 % with Kopek user id: 0
Options	☑ Make this offer the default offer ☐ Offer is no longer available (but active for non-expired previous purchases)
	Modify Remove

4.4.4.3 Personal license

An explanation of the personal license model can be found in chapter 2.10.3.3. The KOPEK Payment System is based on personal users creating a KOPEK user account prior to any purchase. The purchased product, which in this case will be one or more licenses, will only be available for the user that purchased that specific product when validation is done against the KOPEK Payment Server. So personal licenses could be considered achieved just by having the KOPEK Payment Server validate the license every time the licensed product is to be accessed or used. Of course, if the user shares the KOPEK user account key information with another user, the license conditions can be violated. A countermeasure for this is of course to increase the level of identification certainty as described in chapter 4.3.4.1.

4.4.4.4 Single computer license

Chapter 2.10.3.4 has a more detailed discussion of the single computer license model.
If it is absolutely necessary to implement a single computer license model with the KOPEK Payment System, either writing the computer's hard drive's serial number to a local encrypted license file or, if an Internet connection is always present, sending it to a central activation server can achieve this. Then the license file or the activation server can be checked as often as this is desired. The method of having an activation server implies a separate licensing system in addition to the KOPEK Payment System, so it is probably not an optimal solution. The KOPEK Payment System will still be important for checking that every license has been charged for. If a local encrypted licensing file is used for storing the computer's hard drive's serial number, the KOPEK Payment Server can be checked before the file is written so that it isn't written at all if the license hasn't been purchased. In the same way, before the serial number is sent to a central activation server the KOPEK Payment Server could validate the license. But in both cases the process could also be based on the KOPEK Payment Server not allowing the serial number to be submitted if there isn't registered a valid payment transaction. In the case of an encrypted licensing file, it is probably a good idea to have the generation of the file take place on a central and secure server and not on the customer's computer anyway. This in order to make the encryption of the file harder to break for the customer so abuse can be avoided.

Figure 4.20: Single computer license system

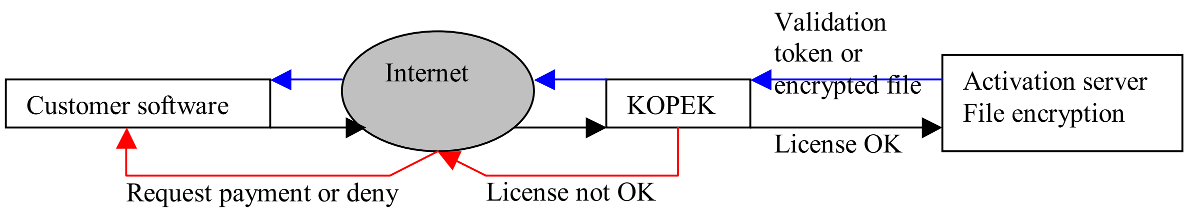


Figure 4.20 shows how a single computer licensing system can be implemented using the KOPEK Payment System. The customer's software sends a request (black arrows) to the activation server for a token or an encrypted file that can unlock the software which otherwise is locked for use or limited in functionality. The request has to pass through the KOPEK Payment Server (KOPEK), and is only allowed to reach the activation server if a valid license has been purchased. If the license isn't valid a request for payment or an error message is returned to the customer's software (red arrows). But if the license was valid, the validation token or an encrypted file will be returned to the customer's software (blue arrows). The customer's software verifies that the activation token or encrypted file is valid and unlocks or

opens all purchased functionality. Of course, the validation token or the encrypted file can contain information about which specific modules of the customer’s software that have been purchased and for how long the licenses are valid.

4.4.4.5 Per user license

The per user license model is outlined in chapter 2.10.3.5 and is easily implemented using the KOPEK Payment System. The software will like in figure 4.20 send a request to a web server behind the KOPEK Payment Server which verifies that the license is valid for the user logged on to the computer where the software is run. In turn the web server will sign the request and return it to the software so the signed request can be verified. The reason for the signing of the request and the verification of the signed request by the software is to ensure that it is the correct KOPEK Payment Server the license is approved by and not some fake server. The license model is very similar to the personal license model, so of course this is the same as what is described in chapter 4.3.4.3. But if the license is to be used as a discount license type for more than one user within the same corporation it could seem less obvious how to achieve this just by what has been described so far. Of course, it is possible to manually create a special offer for each specific KOPEK user that is applicable for a discounted license, but that will be very unpractical if the number of users is more than very limited. For client-server based software this could be solved by having different licenses for the server part of the software than the client software. If the software is module based, the server part could require the customer to buy each separate module and then the client part could require just one discounted type of license in order to gain access to all the modules purchased with the server software. But if the software isn't client-server based, some challenges arise when the KOPEK Payment System is to be used for license control. The same challenges also apply if the client part of the software is to be charged per module, but still discounted when more than one user within the corporation is to purchase a license. These challenges can perhaps be summarized with the phrase "automatic quantum discount for more users" which doesn't seem to be supported by the KOPEK Payment System.

4.4.4.6 Per client license

Chapter 2.10.3.6 contains further information about the per client license model in general. Like the per user license, this license model also offers challenges for the KOPEK Payment System with regards to discount for a larger quantum of licenses within a corporation if the software is stand-alone on each client. But if no discount is to be offered for additional licenses, there are still more issues with implementing the KOPEK Payment System as a license control system for this license model on client only software. These are the same issues as discussed in chapter 4.3.4.4 "single computer license". With client-server based software however, the advantage of using the KOPEK Payment System is apparent. Since there often are restrictions that only allows one person or only selected persons to purchase software licenses, making the purchase in connection with the server installation seems more practical. The client licenses can be sold as 'modules' to the server application, so that more simultaneous users are allowed to connect to the server depending on how many 'client modules' that are bought. The client modules can of course exist in varieties of one client only and more clients that are further discounted per client. The only disadvantage with the KOPEK Payment System for this type of use is that the solution requires the KOPEK user that purchased the licenses to log on each time the server software starts. The client software however, can be started without any use of the KOPEK Payment System if this is undesirable.

4.4.4.7 Per processor license

This license model is discussed in chapter 2.10.3.7 and it is simply a variant of the previously discussed per user or per client license model with an unlimited number of client connections. The KOPEK Payment System supports having a special offer, which can include an unlimited client connection license, and that would cover this license model.

4.4.4.8 Academic/discount license and license programs

Since the discussion on this in chapter 2.10.3.8 generally concludes that these are just discounted licenses based on the previously debated license models, it will of course be possible to add special offers for individual KOPEK user accounts or for everyone that would potentially match the criteria of the license. Academic licenses or licenses that are part of some other license program have so far not had any electronic way of checking that the customer is eligible for that license, and this will still be the case with the KOPEK Payment System. An electronic verification of eligibility would require a higher level of identification certainty and a central register of who are eligible to which licenses.

4.4.4.9 Rented license

Based on the discussion of rented licenses in chapter 2.10.3.9, most rented licenses should be handled the same way per user or per client licenses are handled, perhaps with different pricing per user or client. So this license type is supported in the same manner as discussed in chapter 4.3.4.5 and 4.3.4.6. But like the academic or discounted licenses discussed in chapter 4.3.4.8 there are challenges when an electronic enforcement of the correct license type is required. So if the software publisher uses several license models with more than one price model, the customers has to be trusted to buy the correct license types.

4.4.4.10 GNU General Public License (GPL)/Open license

From the description of these licensing models made in chapter 2.10.3.10 one can derive that a payment system for covering distribution costs or selling access to a support service will be sufficient. Since the KOPEK Payment System is excellent for charging for access to files or services, these license models are fully supported.

4.5 Hardware key

The KOPEK Payment System supports hardware keys in the sense that it supports using the BuyPass and BankID identification tokens, which are unique hardware tokens that can be required for access to purchase and use software using the KOPEK Payment System as a licensing solution. However, this solution is at the moment only available for Norwegian users, since these are national solutions for electronic identification. As support for other international solutions or other nations solutions for this are discovered and developed by KOPEK this limitation will expand correspondingly.

4.6 Payment considerations

Most of the software sold today can be purchased with a period of credit from the software vendor. Usually a bill is issued from the software vendor at the time of delivery of the software package. Using an electronic payment system for licensing and selling software has the advantage of having the payment done at the time of delivery or at the time of first use of the software package. Billing has perhaps mainly been used for practical reasons because software vendors don't want their customers to have to pay up-front or in cash at delivery. But it is also "required" by many customers that a period of time after delivery will be given before the payment is due. So even if an electronic payment system is to be used, the possibility of sending bills and allowing the customer to use the software without having received settlement until the payment is due. But traditionally billing has had the disadvantage of not being able to undo or retract the already delivered software package from the customer if settlement hasn't arrived after the period of credit has expired. It can be expensive to have to use formal and legal ways to eventually receive some settlement if there is anything left if the customer has gone bankrupt. This process should of course be much easier when using an electronic licensing system where the license can be retracted so the software is rendered useless for the customer if settlement cannot be achieved.

4.7 KOPEK Payment System vs. other licensing systems

Table 4.1: Comparison of the licensing solutions in this report

Feature	Macrovision FLEXnet Publisher*	SafeNet Sentinel RMS**	SWREG***	KOPEK Payment System
Time limited license	Yes	Yes	No	Yes
Demo/trial licenses	Yes	Yes	Yes	Yes
Personal license	Yes	Yes	Yes	Yes
Single computer license	Yes	Yes	No	No
Per user license	Yes	Yes	No	No
Per client license	Yes	Yes	No	No
Per processor	Yes	Yes	No	No
Discounted license	Yes	Yes	Yes	Yes
Rented license	Yes	Yes	No	Yes
GPL/Open source license	No	No	Yes	Yes
Payment handling	No	No	Yes	Yes
ERP/CRM integration	Yes	Yes	Yes	Yes
Hardware key option	Yes	Yes	No	Yes
Support for modules	Yes	Yes	Yes	Yes
Wrapper license check	Yes	Yes	Yes	Yes
Source code license check	Yes	Yes	No	Yes
Detached licenses	Yes	Yes	No	No
Grace period license	No	Yes	No	Yes
Local license server	Yes	Yes	No	No
Local license client	No	No	No	Yes
Centralized license control	Yes	Yes	No	Yes

The options in the above table only apply to what is directly supplied by the system, not what can be accomplished through using complementary systems.

* According to published product information at the Macrovision web site (R.3)
** According to published product information at the SafeNet web site (R.6)
*** According to published product information at the SWREG web site (R.7)

4.8 Summary

The KOPEK Payment System has some challenges with certain license models. Most of them could be overcome by developing completion modules such as a license activation server solution. But if the enforcement of the correct licensing model for the different customers isn't the most important issue, the KOPEK Payment System offers a great variation of possibilities for license control. Other systems claim to support many options of licensing, but it is hard to verify what combinations of options that are really supported and how much effort it takes to implement the different options when no such information is publicly available.

5 Reliability and stability issues

5.1 Introduction

Any electronic licensing solution for software has to affect the customers and users of the software as little as possible. This also applies to the reliability and stability of their software. Today software is used in critical operations that could cause serious economic damage if the software stops responding, and a licensing solution implemented in the wrong way could cause this. In this chapter issues that may cause software using the KOPEK Payment System as a licensing solution to fail are discussed.

5.2 Consequences if KOPEK AS goes out of business

If KOPEK AS should go out of business, the central KOPEK Payment System servers could also disappear. This would be fatal as the whole system relies on these servers to be on the Internet at any time. Licenses that are purchased for offline use will of course be unaffected, but all other license providers would have to make an arrangement to replace their KOPEK Payment Server so that the customers with legally acquired licenses can continue to use their software. By simply removing the KOPEK Payment Server, all users would gain free access to all licenses, so this can be a temporary solution until a better replacement for the KOPEK Payment Server can be found.

5.3 Consequences if the software publisher goes out of business

Since the software publisher typically would possess the KOPEK Payment Server and the scripts that verifies all licenses it probably would be fatal if the software publisher goes out of business. Without the server and the script for license verification, the software would in the worst-case scenario terminate while running as soon as either the KOPEK Payment Server or the script disappears from the Internet. In the best-case scenario the software would continue to run until the license expires if the license is an offline type of license. The best case for licenses that are online would be that the software simply can't be started, but that already running software will continue to run.

5.4 Internet connection failure

That a company looses it's Internet connection will cause the software using the KOPEK Payment System as a license control system to be affected in exactly the same ways as mentioned in the previous two chapters. So it will be crucial for any critical software system that there are redundancy and automatic fail-over systems that ensures an Internet connection at all times. Several critical software systems today already have this requirement, so it is possible to achieve this. However for small companies with low margins this can be something they can't afford. And it makes the software publisher's product more expensive than it otherwise would be for the customers, so this will definitely not be a competition advantage.

5.5 Summary

The KOPEK Payment System has some vulnerability issues when it comes to licensing solutions, such as the fact that the license control of the application would depend on a stable Internet connection. In addition, both the KOPEK Payment Systems centralized servers and the KOPEK Payment Server used for license control would have to be available in order for the license control system to work. Since ever more applications and people are getting more dependent on a stable Internet connection, this part will probably be more acceptable in the future. But the dependence on the KOPEK Payment System central servers and the KOPEK Payment Server where the license control is performed will probably be harder for customers to accept.

6 Suggested Improvements and Prototype Implementation

6.1 Suggested improvements

6.1.1 Support for more license models

To make the KOPEK Payment System better suited for software publisher's to use as a licensing solution, an easily understandable implementation description and samples should be provided as part of the systems documentation. In addition it would be preferable if the system could support more of the most common licensing models and the possibility for offline licenses. However, that kind of expansion wouldn't be required in order to start using the system for license control if a more straightforward licensing policy is to be used. Since license control is done the same way anywhere in an applications source code, later changes in license policy would be easy to implement without any source code changes. The exception is offline licenses, where the license control would have to be done against a local license file or some other local license token. But this wouldn't require changes to all the license check function calls in the source code, just a modification in the license check function itself.

6.1.2 ERP- and/or CRM-system integration

The biggest challenge for a software publisher would perhaps be to keep updated price information for the different customers. Very often, each customer has negotiated own discount agreements according to the size of the customer's order, whether the customer is part of a pilot-programme or simply performs as an exceptional reference in marketing the software. In addition, there may be resellers or distributors of the software that are to have their share of the revenue. That will also very often be an agreed percentage for each product they will be selling. All this kind of information should be available to the sales department at the software publisher, so an easy integration with ERP- or CRM-systems is perhaps one of the challenges that really would give the KOPEK Payment System an advantage in the competition against other license control systems. The sales departments should easily be able to change the terms for every customer, reseller or distributor as this becomes necessary in order to make new sales. In the same way, the reseller or distributors will want the ability to adjust their revenue in some hard negotiations between the seller and the customer so that sales isn't lost because the software publisher won't go as low (or high) as necessary to make the sale.

6.1.3 Reliability improvements

In order to use the KOPEK Payment System as a license control system, measures must be made in order to assure that the controlled software will continue to operate even though KOPEK AS goes out of business or the central KOPEK Payment System servers becomes unavailable. This also goes for the software publisher and the servers under the software publisher's control that can affect the software licenses controlled by the KOPEK Payment System. The only measure imaginable in order to achieve this is either redundancy of all the key server systems in the solution in such a way that independent companies who go out of business won't affect any licenses, or to have an offline license file containing the latest license information available at all times.

6.2 Prototype implementation

One of the challenges in using the KOPEK Payment System as a licensing solution for software is for the software publishers to be able to maintain control over what has been purchased by which customers. A simple solution to keep track of this would be to have a simple registration form submitted as the initial purchase of the license is done. Such an implementation will be implemented in this report to demonstrate how the KOPEK Payment System can be used as a basis for developing more suitable license control systems. There are greater challenges facing the KOPEK Payment System in order for it to be a complete licensing system for commercial software, but implementing such prototypes would exceed the scope of this report.

6.2.1 Web form user registration and license information recording

A typical web registration form can be used in order to register who a customer is the first time that customer accesses the license control web page. As suggested in chapter 4.3, a PHP-script can be used to verify that the communication between the end user client and the KOPEK Payment Server that serves as a central license control server is secure. By modifying that script, it can be used as a registration form as well.

In addition, the script can register what license modules a KOPEK user has accessed and when the license for these modules expires. This is due to the fact that the KOPEK Payment Server passes this kind of information to the web server as headers. Relevant headers used in the sample script here are:

- HTTP_X_KOPEK_IDENTITY, which contains the user ID of the KOPEK end user
- HTTP_X_KOPEK_EXPIRETIME, which contains the UNIX timestamp for when the license expires. The UNIX timestamp represents the time in the number of seconds from January 1st 1970.
- HTTP_X_KOPEK_CLIENTIP, which contains the IP address from where the KOPEK end user client runs

But the script can only record information, it will not be able to alter licenses that are controlled by the KOPEK Payment Server. In the following sample script, recording is done to a MySQL database. Of course this information could just as well have been written to whatever would be preferred.

Another usage of this script is to make it write some of the relevant information as output text, so it can be used by the application it was called from. That will for instance make it possible for the application to terminate once the license expires without having to poll the KOPEK Payment Server continuously, since the application could read that information during the initial contact it makes to the KOPEK Payment Server.

In the same manner it would of course also be possible for the script to generate an encrypted license string or a part of such a string that represents the currently accessed license module and have that output to the application as well. The application would then have to be able to write that string to a local file, which in turn could be read and decrypted whenever necessary. Hence the challenge of having an offline license solution could be solved in this manner. As the application would know whenever a license or a license module expires the application could also give the end user notice some time before the license expires. An end user would be able to pay for an extended license period before the software seizes to work.

The following is a modified version of this PHP-script, which performs such registration and recording tasks. The sample database used with this sample script can be created using the SQL code in appendix D.

Modified version of getlicense.php:

```
<?php

header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
header("Cache-Control: no-store, no-cache, must-revalidate");
header("Cache-Control: post-check=0, pre-check=0", false);
header("Content-type: text/html");
header("Pragma: No-cache");

require_once('customercheck.php');

assert(get_magic_quotes_gpc());
$challenge = $_GET['challenge'];

if((strlen($challenge) < 5) ||
   (strlen($challenge) > 512))
{
    echo "Error: Invalid challenge";
    echo "<BR>Challenge: ".$challenge;
    exit;
}

if(strlen($challenge) > 0)
{
   // If a new customer has been submitted for registration, the registration
   // will be handled here.
   if($_POST['submitted'] == 1)
   {
      create_customer($_POST['kopekid'], $_POST['customername'], $_POST['contactname'],
$_POST['email']);
   }
   // If the customer exists in the database, proceed with normal response to
   // the challenge that was presented in the web request.
   // If the customer doesn't exist in the database, a form presenting field
   // in which a customer can register will be displayed.
   if($_SERVER[HTTP_X_KOPEK_IDENTITY] == 0 || find_customer($_SERVER[HTTP_X_KOPEK_IDENTITY]))
   {
      // If the customer hasn't accessed this license module before, the
      // use of the license module will be registered before proceeding
      if($_SERVER[HTTP_X_KOPEK_IDENTITY] <> 0 &&
!find_customer_license($_SERVER[HTTP_X_KOPEK_IDENTITY], $_GET['productid'],
$_SERVER[HTTP_X_KOPEK_EXPIRETIME]))
      {
         create_customer_license($_SERVER[HTTP_X_KOPEK_IDENTITY], $_GET['productid'],
$_SERVER[HTTP_X_KOPEK_EXPIRETIME], $_SERVER[HTTP_X_KOPEK_CLIENTIP]);
      }
      $tempFile = tempnam("/tmp", "challenge");

      $handle = fopen($tempFile, "w");
      fwrite($handle, substr($_SERVER['REQUEST_URI'], strpos($_SERVER['REQUEST_URI'],
"challenge=") + 10));
      fclose($handle);
      echo "KOPEKUID: " . $_SERVER[HTTP_X_KOPEK_IDENTITY] . "\r\n<BR>\r\n";
      echo "ExpireTime: " . $_SERVER[HTTP_X_KOPEK_EXPIRETIME] . "\r\n<BR>\r\n";
      echo "Signature: " . exec("/bin/cat \"" . $tempFile  . "\" | /usr/bin/openssl sha1 -sign
license.key | /usr/bin/openssl base64 -e | /usr/bin/awk '{ printf $0 }'") . "\r\n<BR>\r\n";
      unlink($tempFile);
   }
   else
   {
      // Register that this license module has been accessed by this customer
      // no matter if the customer enters the registration information or not.
      if(!find_customer_license($_SERVER[HTTP_X_KOPEK_IDENTITY], $_GET['productid'],
$_SERVER[HTTP_X_KOPEK_EXPIRETIME]))
      {
         create_customer_license($_SERVER[HTTP_X_KOPEK_IDENTITY], $_GET['productid'],
$_SERVER[HTTP_X_KOPEK_EXPIRETIME], $_SERVER[HTTP_X_KOPEK_CLIENTIP]);
      }
      display_customerform($_SERVER[HTTP_X_KOPEK_IDENTITY], $_GET['productid']);
   }
}
?>
```

The script file containing the functions used in getlicense.php, customercheck.php, is included in getlicense.php and it contains the following lines of code:

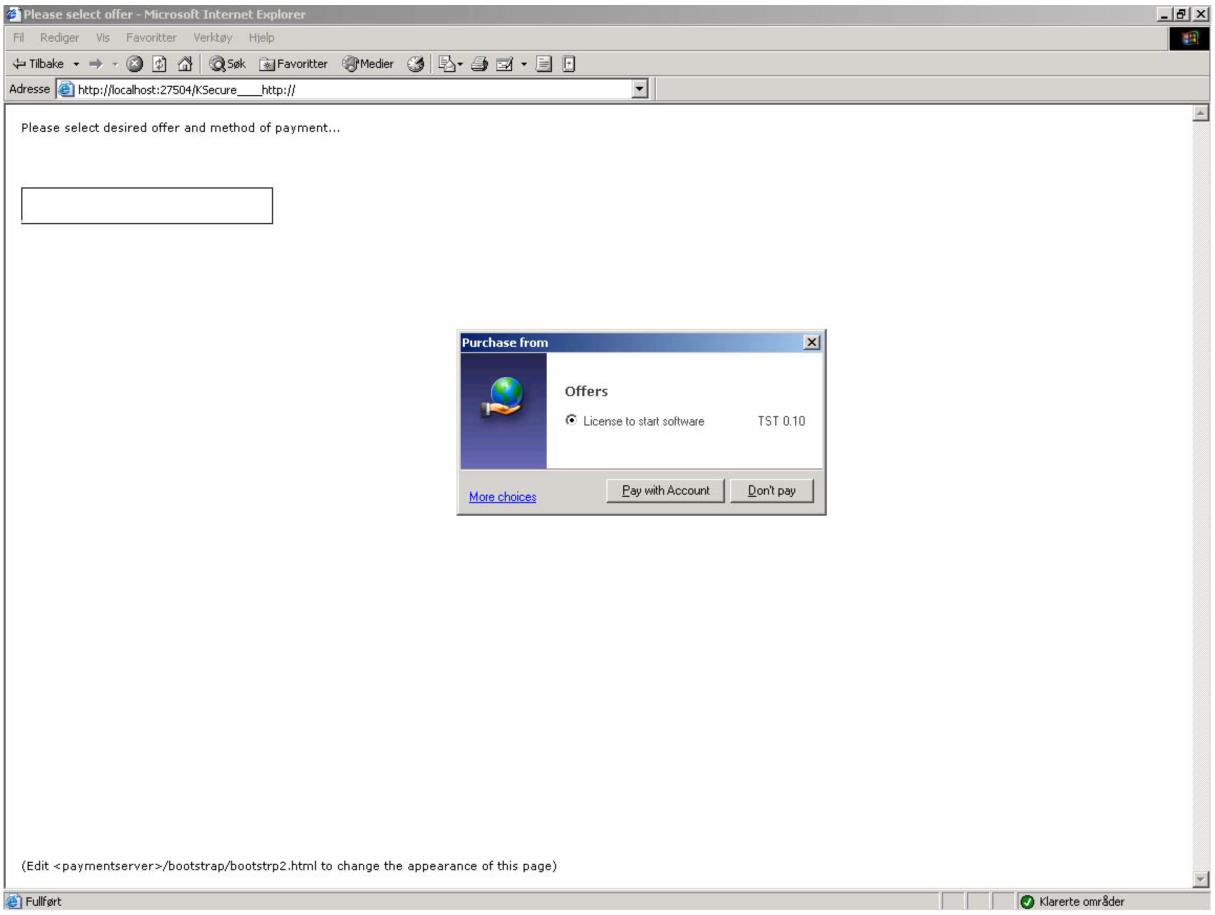
```
<?php

// Function to connect to the database where the customer information resides.
function connect_DB()
{
   // Connect to the database server locally with the username webreader
   // and a blank password
   $link = mysql_connect("localhost","webreader","");
   if(!$link)
   {
      die("Unable to connect to database server: ".mysql_error());
   }

   // Select the CustomerDB database where the information in question
   // resides.
   mysql_select_db("CustomerDB", $link) or die("Unable to connect to customer database:
".mysql_error());
}

// Function that when given an existing customer's KOPEK ID will
// return true, when given a new customer's KOPEK ID will return
// false.
function find_customer($kopekid)
{
   connect_DB();
   // Query the database table for whether or not the customer already
   // exists in the database
   $sqlquery = "select distinct customers.id as customerid from customers where
kopekid='".$kopekid."'";
   $result = mysql_query($sqlquery) or die("Could not execute query:
".mysql_error()."<BR>\r\nSQL: ".$sqlquery);
   if(mysql_num_rows($result) > 0)
   {
      $customerid = mysql_result($result, 0, "customerid");
      if($customerid > 0)
      {
         return true;
      }
      else
      {
         return false;
      }
   }
   else
   {
      return false;
   }
   mysql_close();
}
```

```
function find_customer_license($kopekid, $licensemodule, $expiretime)
{
   connect_DB();
   // Query the database table for whether or not the customer has accessed
   // this license module previously
   $sqlquery = "select distinct customerlicense.id as customerlicenseid from customerlicense
where kopekid='".$kopekid."' and licensemodule='".$licensemodule."'";
   if(is_numeric($expiretime))
   {
      $sqlquery .= " and expiretime=FROM_UNIXTIME(".$expiretime.")";
   }
   else if($expiretime == "Never")
   {
      $sqlquery .= " and expiretime=FROM_UNIXTIME(1)";
   }
   else if($expiretime == "One-Time")
   {
      $sqlquery .= " and expiretime IS NULL";
   }
   $result = mysql_query($sqlquery) or die ("Could not execute query:
".mysql_error()."<BR>\r\nSQL: ".$sqlquery);
   if(mysql_num_rows($result) > 0)
   {
      $customerlicenseid = mysql_result($result, 0, "customerlicenseid");
      if($customerlicenseid > 0)
      {
         return true;
      }
      else
      {
         return false;
      }
   }
   else
   {
      return false;
   }
   mysql_close();
}

function create_customer($kopekid, $customername, $contactname, $email)
{
   connect_DB();
   // Registration of a new customer
   $sqlquery = "insert into customers (customername, contactname, email, kopekid) VALUES
('".$customername."', '".$contactname."', '".$email."', '".$kopekid."')";
   $result = mysql_query($sqlquery);
   if(!$result)
   {
      die("Invalid query: ".mysql_error());
   }
   mysql_close();
}
```

```
function create_customer_license($kopekid, $licensemodule, $expiretime, $clientip)
{
   connect_DB();
   // Register that this license module has been accessed by this customer
   // for the first time.
   $sqlquery = "insert into customerlicense (kopekid, licensemodule, datefirstactivated,
expiretime, clientip) VALUES ('".$kopekid."', '".$licensemodule."', NOW()";
   if(is_numeric($expiretime))
   {
      $sqlquery .= ", FROM_UNIXTIME(".$expiretime.")";
   }
   else if($expiretime == "Never")
   {
      $sqlquery .= ", FROM_UNIXTIME(1)";
   }
   else if($expiretime == "One-Time")
   {
      $sqlquery .= ", NOW()";
   }
   else
   {
      $sqlquery .= ", NOW()";
   }
   $sqlquery .= ", '".$clientip."')";   $result = mysql_query($sqlquery);
   if(!$result)
   {
      die("Invalid query: ".mysql_error());
   }
   mysql_close();
}

function display_customerform($kopekid, $productid)
{
   // This function will display a web form in which a customer can enter
   // the required registration information. This information can however
   // just as well be entered and submitted through a post to this script
   // from any other registration form.
   echo "<TABLE>\n";
   echo "   <FORM METHOD=\"post\" ACTION=\"".$PHP_SELF."\">\n";
   echo "      <TR>\n";
   echo "         <TD>\n";
   echo "            Customer name:\n";
   echo "         </TD>\n";
   echo "         <TD>\n";
   echo "            <INPUT TYPE=\"text\" NAME=\"customername\">\n";
   echo "         </TD>\n";
   echo "      </TR>\n";
   echo "      <TR>\n";
   echo "         <TD>\n";
   echo "            Contact name:\n";
   echo "         </TD>\n";
   echo "         <TD>\n";
   echo "            <INPUT TYPE=\"text\" NAME=\"contactname\">\n";
   echo "         </TD>\n";
   echo "      </TR>\n";
   echo "      <TR>\n";
   echo "         <TD>\n";
   echo "            E-mail address:\n";
   echo "         </TD>\n";
   echo "         <TD>\n";
   echo "            <INPUT TYPE=\"text\" NAME=\"email\">\n";
   echo "         </TD>\n";
   echo "      </TR>\n";
   echo "      <TR>\n";
   echo "         <TD COLSPAN=\"2\">\n";
   echo "            <INPUT TYPE=\"hidden\" NAME=\"kopekid\" VALUE=\"".$kopekid."\">\n";
   echo "            <INPUT TYPE=\"hidden\" NAME=\"challenge\" VALUE=\"".$challenge."\">\n";
   echo "            <INPUT TYPE=\"hidden\" NAME=\"submitted\" VALUE=\"1\">\n";
   echo "            <INPUT TYPE=\"hidden\" NAME=\"productid\" VALUE=\"".$productid."\">\n";
   echo "            <INPUT TYPE=\"submit\" VALUE=\"OK\">\n";
   echo "         </TD>\n";
   echo "      </TR>\n";
   echo "   </FORM>\n";
   echo "</TABLE>\n";
}
```

The usage of the above sample PHP script is demonstrated in figures 6.1-6.3.

Figure 6.1: KOPEK Payment Client purchase offer


Please select offer - Microsoft Internet Explorer
Fil Rediger Vis Favoritter Verktøy Hjelp
Tilbake Søk Favoritter Medier
Adresse http://localhost:27504/KSecure____http://
Purchase from
Offers
More choices
Pay with Account
Don't pay
Fullført
Klarerte områder

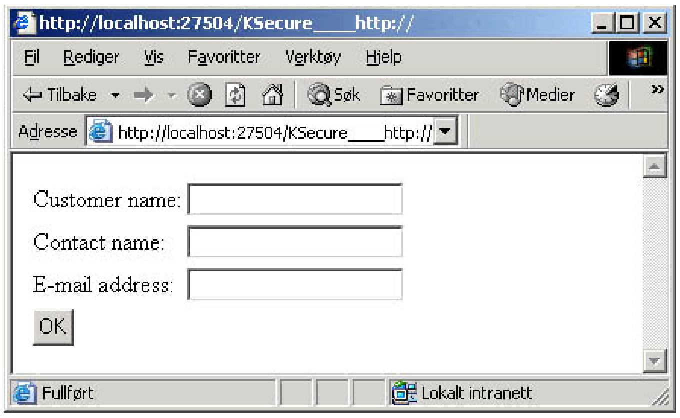
Please select desired offer and method of payment...

License to start software TST 0.10

(Edit <paymentserver>/bootstrap/bootstrp2.html to change the appearance of this page)

The offer to purchase the license for the requested software module will appear in front of web page generated by the PHP script.

Figure 6.2: Web form displaying fields for registration of new customer information


http://localhost:27504/KSecure____http://
Fil Rediger Vis Favoritter Verktøy Hjelp
Tilbake Søk Favoritter Medier
Adresse http://localhost:27504/KSecure____http://
OK
Fullført
Lokalt intranett

Customer name:

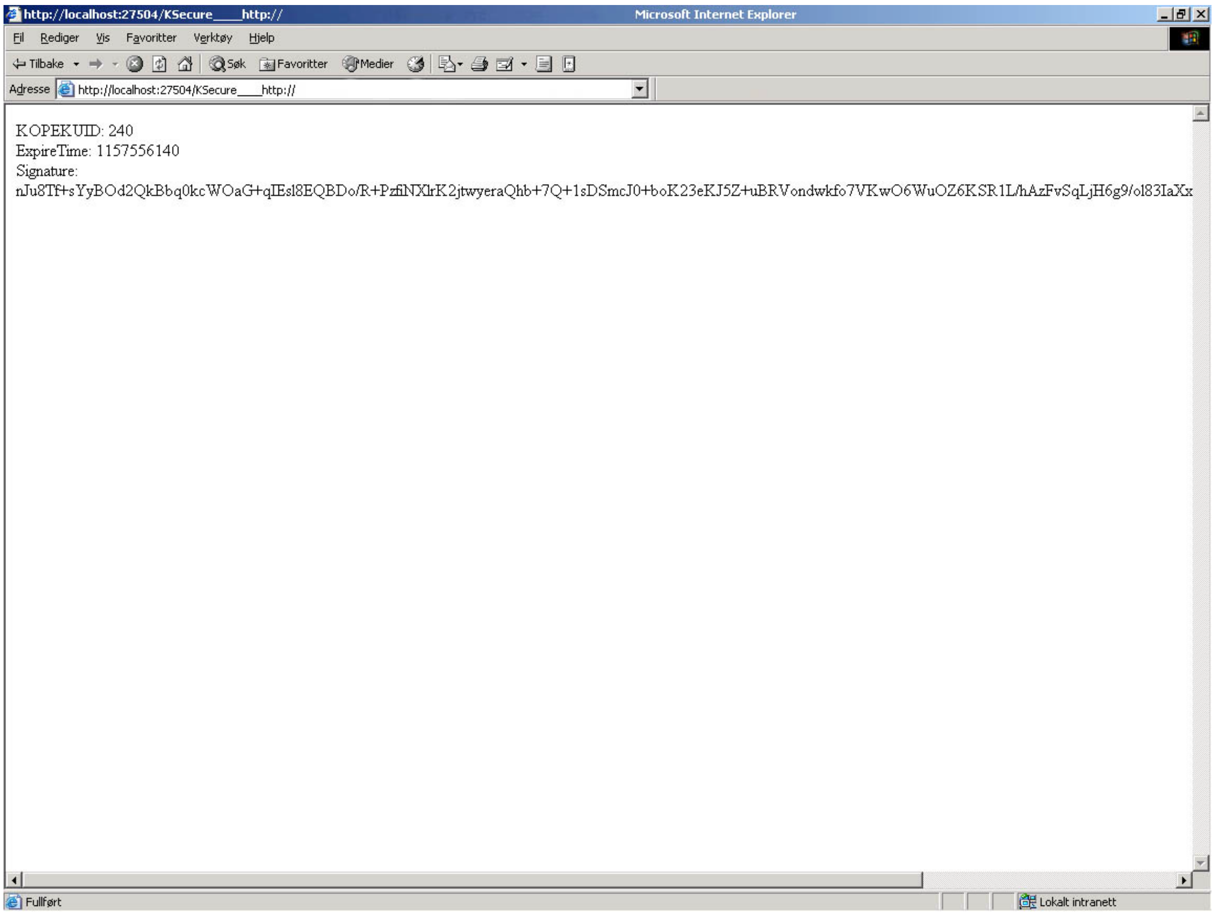
Contact name:

E-mail address:

After having posted the customer information form when KOPEK user ID 240 was logged in the entered information can be found in the database:

```
mysql> select * from customers;
+----+---------+--------------+-------------+----------------+
| id | kopekid | customername | contactname | email          |
+----+---------+--------------+-------------+----------------+
|  1 |     240 | custname     | contname    | email@test.com |
+----+---------+--------------+-------------+----------------+
```

Figure 6.3: Web page with data feedback for the application


http://localhost:27504/KSecure____http:// Microsoft Internet Explorer
Fil Rediger Vis Favoritter Verktøy Hjelp
Tilbake Søk Favoritter Medier
Adresse http://localhost:27504/KSecure____http://
Fullført
Lokalt intranett

```
KOPEKUID: 240
ExpireTime: 1157556140
Signature:
nJu8Tf+sYyBOd2QkBbq0kcWOaG+qIEsl8EQBDo/R+PzfiNXlrK2jtwyeraQhb+7Q+1sDSmcJ0+boK23eKJ5Z+uBRVondwkfo7VKwO6WuOZ6KSR1L/hAzFvSqLjH6g9/ol83IaXx
```

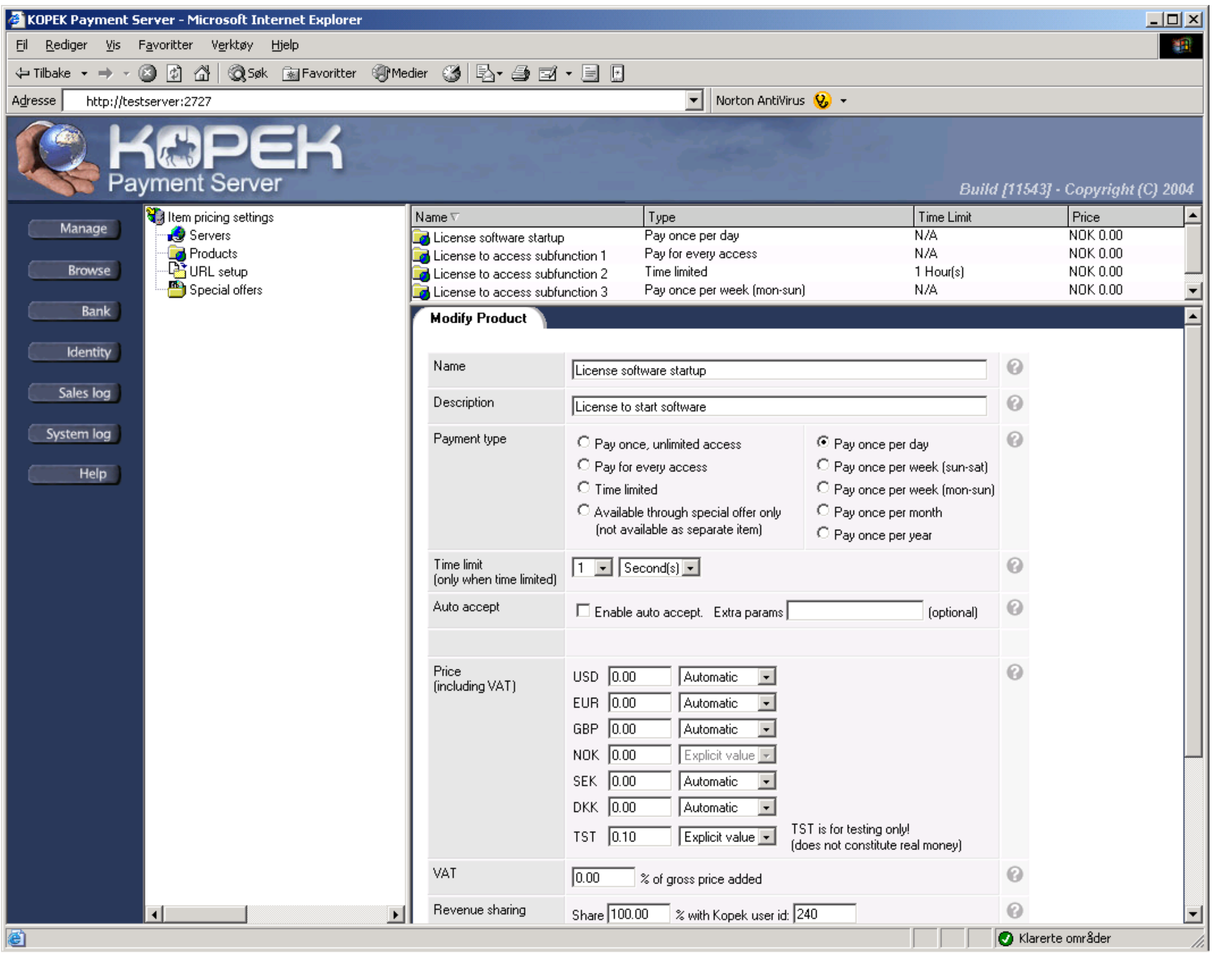
The web page is presenting information and the signature response to the challenge presented to the PHP script by the application. This information can be read by the application so the signature can be verified. The additional information can be used by the application if this is useful.

6.2.2 Software application prototype

A sample application which uses the above PHP-script and illustrates how license control can be achieved using the KOPEK Payment System is detailed in this chapter. The application is using a personal license model (as described in chapter 2.10.3.4) but contains 4 modules that each needs a separate license. The first module is mandatory in order to access the other modules and is called the initial module. The remaining three modules can be accessed after a valid license for the module in question has been purchased.

The license modules have been defined in the KOPEK Payment Server, where the price and time of validity of the license are specified (figure 6.4).

Figure 6.4: Definition of license modules, prices and validity of licenses


KOPEK Payment Server - Microsoft Internet Explorer
Fil Rediger Vis Favoritter Verktøy Hjelp
Tilbake Søk Favoritter Medier
Adresse http://testserver:2727
Norton AntiVirus
KOPEK Payment Server
Build [11543] - Copyright (C) 2004
Manage
Browse
Bank
Identity
Sales log
System log
Help
Item pricing settings
Servers
Products
URL setup
Special offers
Klarerte områder

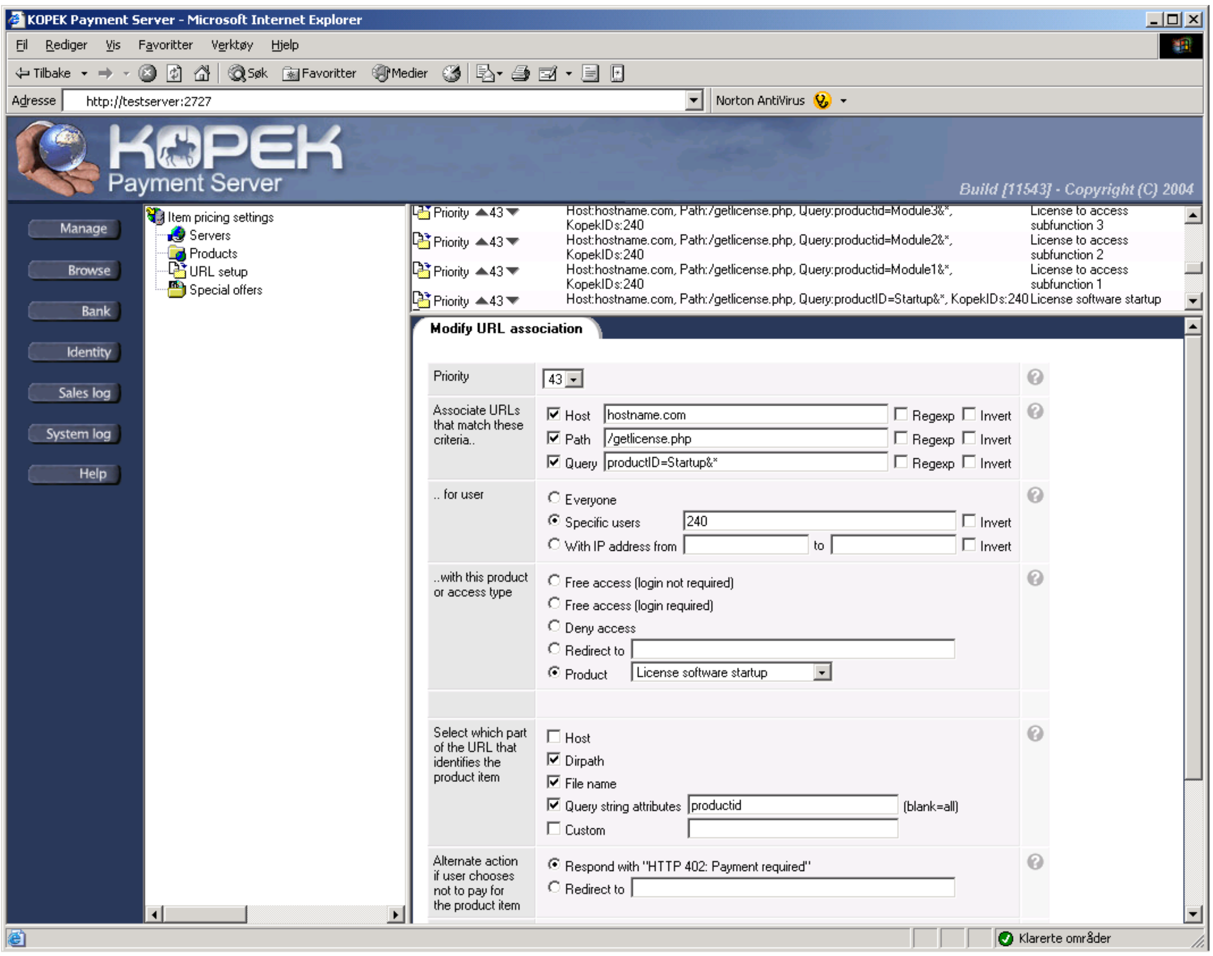
Name	Type	Time Limit	Price
License software startup	Pay once per day	N/A	NOK 0.00
License to access subfunction 1	Pay for every access	N/A	NOK 0.00
License to access subfunction 2	Time limited	1 Hour(s)	NOK 0.00
License to access subfunction 3	Pay once per week (mon-sun)	N/A	NOK 0.00

Modify Product

Name	License software startup
Description	License to start software
Payment type	Pay once, unlimited access / Pay for every access / Time limited / Available through special offer only (not available as separate item) / [✓ Pay once per day] / Pay once per week (sun-sat) / Pay once per week (mon-sun) / Pay once per month / Pay once per year
Time limit (only when time limited)	1 Second(s)
Auto accept	Enable auto accept. Extra params (optional)
Price (including VAT)	USD 0.00 Automatic EUR 0.00 Automatic GBP 0.00 Automatic NOK 0.00 Explicit value SEK 0.00 Automatic DKK 0.00 Automatic TST 0.10 Explicit value — TST is for testing only! (does not constitute real money)
VAT	0.00 % of gross price added
Revenue sharing	Share 100.00 % with Kopek user id: 240

As shown in figure 6.4, the license for the initial module which allows the end user to start the application is defined to be valid for just the same day it was purchased. The price is set to 0.10 TST (the testing currency). The license to access sub function 1 or Module 1, which it is named in the application, is defined to only be valid for one time. Module 1 will cost 0.01 TST, and the end user has to purchase this license every time Module 1 will be accessed or started. Once started, the module can run indefinitely. Sub function 2 or Module 2 will also cost 0.01 TST, but it valid for one hour from the time it was purchased. It doesn't necessarily mean that Module 2 will shut down after one hour (it could however be implemented in that way as well), but it means that Module 2 can be run as many times the user wants within an hour. Module 3 or sub function 3 will cost 0.01 TST and the license is valid within the current week, that is it can be run as many times the end user wants from Monday trough Sunday.

Figure 6.5: Linking URLs with product



KOPEK Payment Server - Microsoft Internet Explorer

Adresse http://testserver:2727

Build [11543] - Copyright (C) 2004

Item pricing settings
- Servers
- Products
- URL setup
- Special offers

Priority 43	Host:hostname.com, Path:/getlicense.php, Query:productid=Module3&*, KopekIDs:240	License to access subfunction 3
Priority 43	Host:hostname.com, Path:/getlicense.php, Query:productid=Module2&*, KopekIDs:240	License to access subfunction 2
Priority 43	Host:hostname.com, Path:/getlicense.php, Query:productid=Module1&*, KopekIDs:240	License to access subfunction 1
Priority 43	Host:hostname.com, Path:/getlicense.php, Query:productID=Startup&*, KopekIDs:240	License software startup

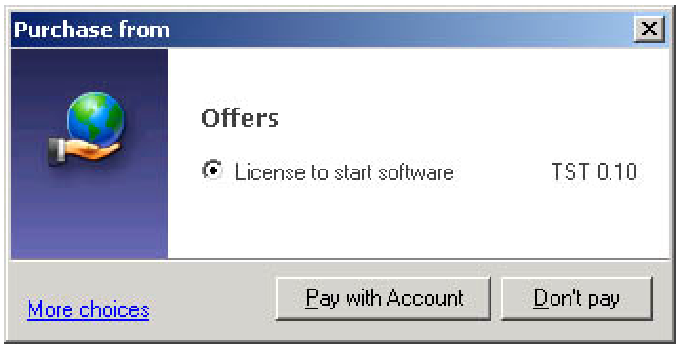
Modify URL association

Priority	43
Associate URLs that match these criteria..	[✓] Host: hostname.com ☐ Regexp ☐ Invert [✓] Path: /getlicense.php ☐ Regexp ☐ Invert [✓] Query: productID=Startup&* ☐ Regexp ☐ Invert
.. for user	○ Everyone ● Specific users: 240 ☐ Invert ○ With IP address from to ☐ Invert
..with this product or access type	○ Free access (login not required) ○ Free access (login required) ○ Deny access ○ Redirect to ● Product: License software startup
Select which part of the URL that identifies the product item	☐ Host [✓] Dirpath [✓] File name [✓] Query string attributes: productid (blank=all) ☐ Custom
Alternate action if user chooses not to pay for the product item	● Respond with "HTTP 402: Payment required" ○ Redirect to

Figure 6.5 shows how the end user can be offered the correct license as the application tries to access the PHP script with only the productid parameter specifying which module the end user is trying to access. For more information on how the configuration of the KOPEK Payment Server is done, see chapter 4.4.3.

This result in the dialogs and screenshots following in figure 6.6-6.14 that the end user will see when trying to access the different modules:

Figure 6.6: Offer in order to start the application

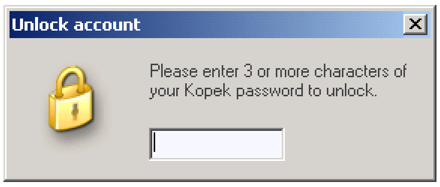


Purchase from

Offers

● License to start software TST 0.10

Figure 6.7: Unlock KOPEK Client application

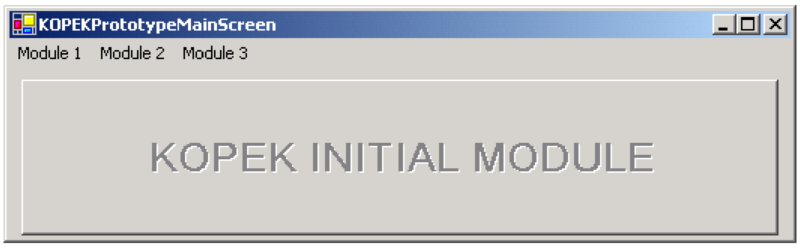


Please enter 3 or more characters of your Kopek password to unlock.

If the KOPEK Client hasn't been used for a while, the account has to be unlocked (figure 6.7).

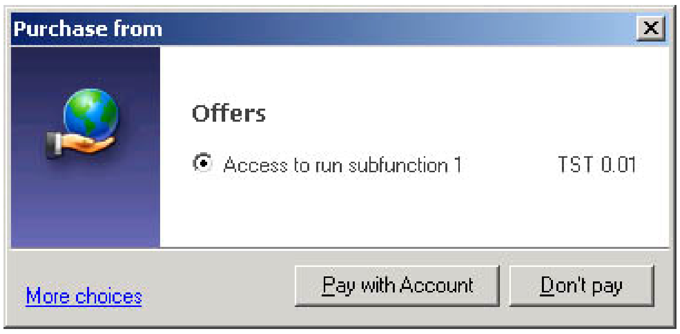
After payment has been received, the application is started (figure 6.8).

Figure 6.8: Initial module of the prototype



Trying to access Module 1, the end user is required to purchase a valid license for that module (figure 6.9).

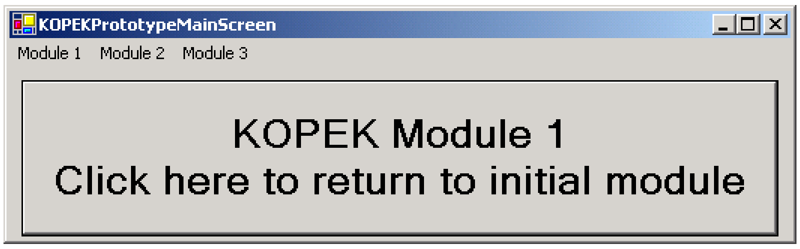
Figure 6.9: Purchase offer for Module 1



● Access to run subfunction 1 TST 0.01

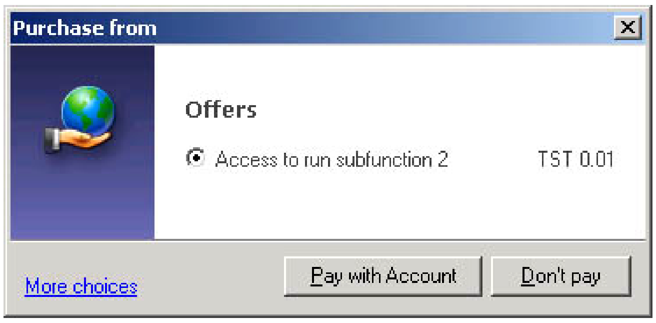
When access to Module 1 is granted, the end user is allowed to use Module 1 indefinitely, but if the end user goes from Module 1 to another module of the software, the user will once again be prompted to pay if the end user tries to access Module 1 once again.

Figure 6.10: Module 1



Trying to access Module 2 for the first time, the end user has to purchase that license as well (figure 6.11).

Figure 6.11: Offer to purchase license for Module 2



Access to Module 2 is granted and can be accessed indefinitely as long as the end user doesn't leave Module 2. But if the end user leaves Module 2, the end user can still access Module 2 if this is done within the hour the license is valid for.

Figure 6.12: Module 2

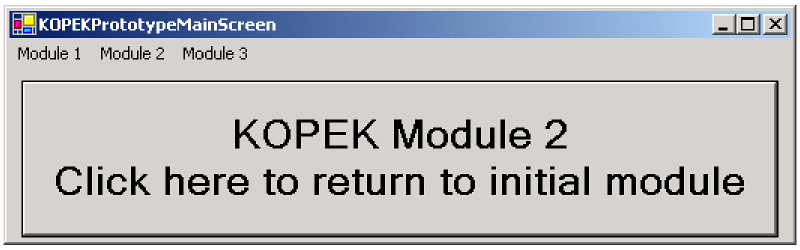
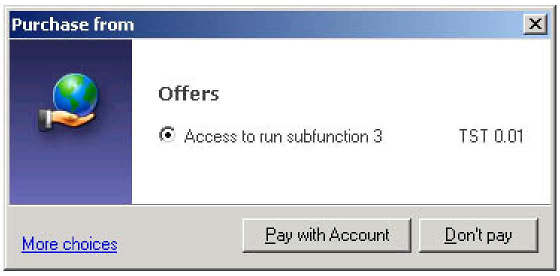
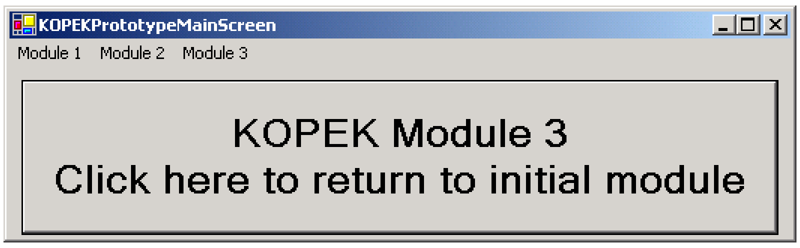


Figure 6.13: Offer to the end user in order to access Module 3



After a license to access Module 3 has been purchased, the user can use Module 3 indefinitely and the end user can access Module 3 as many times as the end user wants within the week for which the license is valid.

Figure 6.14: Module 3



As these modules are accessed and licenses purchased, this will be recorded in the MySQL database that the PHP script accesses each time it is invoked. The following shows what has been recorded from when the application was first started. The end user has been accessing Module 1 twice and Module 2 and Module 3 several times within an hour of the initial purchase.

```
mysql> select * from customerlicense;
+----+---------+---------------+---------------------+---------------------+--------------+
| id | kopekid | licensemodule | datefirstactivated  | expiretime          | clientip     |
+----+---------+---------------+---------------------+---------------------+--------------+
|  1 |     240 | Startup       | 2006-09-13 15:03:33 | 2006-09-13 23:59:59 | xx.xxx.x.xxx |
|  2 |     240 | Module1       | 2006-09-13 15:12:31 | 2006-09-13 15:12:31 | xx.xxx.x.xxx |
|  3 |     240 | Module1       | 2006-09-13 15:13:42 | 2006-09-13 15:13:42 | xx.xxx.x.xxx |
|  4 |     240 | Module2       | 2006-09-13 15:19:32 | 2006-09-13 16:20:47 | xx.xxx.x.xxx |
|  5 |     240 | Module3       | 2006-09-13 15:32:19 | 2006-09-17 23:59:59 | xx.xxx.x.xxx |
+----+---------+---------------+---------------------+---------------------+--------------+
```

The expire time recorded is as expected, Module 1 expires at the same time as it was purchased, Module 2 expires one hour after it was purchased and Module 3 expires at midnight the following Sunday.

Of course, the above time limits are only useful for time limited trial versions of an application or for making a point about how the time limitations work in such an example. If a serious application would implement time limits, it probably would be more practical with one year and not one hour or one week as a validity time frame.

The source code for this prototype application can be found in appendix E. Notice that the KOPEKFunctions class has been modified from the version presented in chapter 4.3 so that it posts static customer information to the PHP script if the web form asking for customer information is presented to the application. This could of course just as well be information retrieved by the application from the end user.

6.3 Summary

Some possible improvements in order to make the KOPEK Payment System more suited to be a licensing system for commercial software are outlined in this chapter. Of these suggestions, some would require hardware improvements while others are more dependent on further software implementations.

The prototypes detailed in this chapter shows how some of the simpler licensing models can be handled by using the KOPEK Payment System as a foundation for license control. It is also demonstrated how more extensive customer and end user information can be collected.

Still, the more advanced licensing models would require the writing of almost an entirely new license control system in order for that to be supported by the KOPEK Payment System. This would be far beyond the scope of this report and hence there is no attempt to make such a prototype here.

7 Summary and Conclusion

7.1 Summary

The KOPEK Payment System supports payment of online content or services as well as single sign in. Setting up the KOPEK Payment System is easy, the system consist of the KOPEK Payment Server as well as an end user client. The server is installed so that it will be exposed to the desired customers, usually directly on the Internet. It can be installed in front an existing web server that handles more advanced content or services than the KOPEK Payment Server itself supports. Access to the content or the service will then be controlled by the rules configured in the KOPEK Payment Server. The KOPEK client application will be used from any end user who will access that content in order to identify the end user and make sure the user has met the requirements defined in the KOPEK Payment Server.

Simple license control over personal licenses can be achieved by creating a web page and make the KOPEK Payment Server control access to that web page. An application can then try to access that web page and will only proceed if the web page can be read. If the web page cannot be read, the end user hasn't met the requirements defined in the KOPEK Payment Server and hence the end user doesn't have a valid license. This results in support for quite advanced copy protection when using the KOPEK Payment System. The copy protection can be further enhanced by also using the KOPEK Payment Systems support for centralized electronic identification systems, like the Norwegian BankID or BuyPass.

In the market there exists some solutions that have specialized in license control and copy protection. These solutions seem to cover most of the more advanced licensing models, but none of these solutions can supply both a complete solution for license control and copy protection as well as a solution for electronic distribution of the licenses.

There would probably be a market for such complete solutions and since the KOPEK Payment System already support both electronic distribution of licenses and can be used for simple license control an expansion of the KOPEK Payment System to also support more advanced license control.

However, the KOPEK Payment System has some obvious weaknesses when it comes to dependence on an Internet connection and that both the company responsible for the KOPEK Payment System central servers as well as the company responsible for the KOPEK Payment Server used for license control can continue to keep the servers operative.

7.2 Conclusion

The KOPEK Payment System is possible to use as a licensing solution with ease for some simpler license models. But it is far from a complete licensing solution that will allow software publishers full flexibility when it comes to the choice of licensing models. It will provide good copy protection when used as an online license system for personal licenses. For personal licenses it also provides the software publishers an opportunity to distribute and sell licenses electronically and safe online. Of course, more advanced features can be developed based on the functionality provided by the KOPEK Payment System. But it would require much more logic than the KOPEK Payment System provides to achieve full support for the more advanced licensing models commonly used today. So the KOPEK Payment System is best used as exactly that, a system for electronic payment. And that includes the possibility of payment for software licenses, but not for the purpose of license control or copy protection.

To achieve a complete licensing solution for software that has support for electronic payment, the easiest way would probably be through existing licensing solution providers that could incorporate the KOPEK Payment System in their solutions.

Appendix A. Table of figures

Figure 3.1: Sentinel RMS overview (source of figure is SafeNet web site, R.6) 23
Figure 4.1: The KOPEK Payment System overview (figure from the KOPEK Payment Server documentation) 27
Figure 4.2: Proxy-type server (figure from the KOPEK Payment Server documentation)......28
Figure 4.3: Parallel proxy-type server (figure from the KOPEK Payment Server documentation) 28
Figure 4.4: Stand-alone server (figure from the KOPEK Payment Server documentation) 28
Figure 4.5: The KOPEK Payment Client end user authentication box 29
Figure 4.6: The KOPEK Payment Client new end user registration form 29
Figure 4.7: KOPEK Payment Client second registration form when creating a new end user 30
Figure 4.8: KOPEK Payment Client final step of new end user registration 30
Figure 4.9: KOPEK Payment Client new site information window 31
Figure 4.10: Simple options available to an end user when access isn't directly granted........31
Figure 4.11: Advanced options available to an end user when access isn't directly granted...31
Figure 4.12: Error message returned when the end user doesn't comply with any of the options 31
Figure 4.13: KOPEK Payment Client deposit money dialog 32
Figure 4.14: KOPEK Payment Client Deposit money dialog using Norwegian currency.......32
Figure 4.15: KOPEK Payment Client deposit money from VISA card dialog 32
Figure 4.16: KOPEK Payment Client account overview 33
Figure 4.17: Configuring KOPEK Payment Server products..39
Figure 4.18: Configuring KOPEK Payment Server filters 40
Figure 4.19: Configuring KOPEK Payment Server special offers...41
Figure 4.20: Single computer license system 42
Figure 6.1: KOPEK Payment Client purchase offer 60
Figure 6.2: Web form displaying fields for registration of new customer information 60
Figure 6.3: Web page with data feedback for the application 61
Figure 6.4: Definition of license modules, prices and validity of licenses.............................62
Figure 6.5: Linking URLs with product..63
Figure 6.6: Offer in order to start the application 63
Figure 6.7: Unlock KOPEK Client application 64
Figure 6.8: Initial module of the prototype..64
Figure 6.9: Purchase offer for Module 1 64
Figure 6.10: Module 1 64
Figure 6.11: Offer to purchase license for Module 2 65
Figure 6.12: Module 2 65
Figure 6.13: Offer to the end user in order to access Module 3..65
Figure 6.14: Module 3 65

Appendix B. Tables

Table 6.1: Comparison of the licensing solutions in this report 48

Appendix C. Table of references

R.1: Andrew S. Tanenbaum, Prentice Hall International 1996: Computer Networks, third edition, international edition. ISBN 0-13-394248-1
R.2: http://www.gnu.org/licenses/gpl.html, Version 2, June 1991 Free Software Foundation
R.3: http://www.macrovision.com/solutions/software/index.shtml
R.4: Macrovision whitepaper "Electronic Licensing: The Build vs. Buy Decision" (Marked FpupmWP.0304, 2005 edition)
(http://mktg.macrovision.com/mk/submit/mvsn_submit?_JS=T&lead_source=FNP_X_X_11_L_FNP11_2005_11_W_WP_BVB_X_X_NAM&dname=Electronic%20Licensing:%20The%20Build%20vs.%20Buy%20Decision&pcode=fnp&target=downloads/products/flexnet_publisher/white_papers/ElectronicLic_BuildBuy.pdf)
R.5: Macrovision datasheet "FLEXnet Publisher Licensing Module" (Marked FPlmDS.0104, 2004 edition)
http://www.macrovision.com/downloads/products/flexnet_publisher/licensing/datasheets/fnp_lm_datasheet.pdf
R.6: http://www.safenet-inc.com/digital_rights_management/rmsv.asp
R.7: http://www.swreg.org

Appendix D. MySQL database definition for sample database

```
insert into db (Host, Db, User, Select_priv, Insert_priv, Update_priv, Delete_priv,
Create_priv, Drop_priv, Grant_priv, References_priv, Index_priv, Alter_priv) values
('localhost', 'CustomerDB', 'webreader', 'Y', 'Y', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N');

insert into user (Host, User, Password, Select_priv, Insert_priv, Update_priv, Delete_priv,
Create_priv, Drop_priv, Reload_priv, Shutdown_priv, Process_priv, File_priv, Grant_priv,
References_priv, Index_priv, Alter_priv) values ('localhost', 'webreader', '', 'Y', 'N', 'N',
'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N');

create database CustomerDB;

FLUSH PRIVILEGES;

use CustomerDB;

create table customers (
   id int unsigned AUTO_INCREMENT NOT NULL,
   kopekid int unsigned NOT NULL,
   customername varchar(100),
   contactname varchar(100),
   email varchar(50),
   PRIMARY KEY(id)
   );

create table customerlicense (
   id int unsigned AUTO_INCREMENT NOT NULL,
   kopekid int unsigned NOT NULL,
   licensemodule varchar(100) NOT NULL,
   datefirstactivated datetime NOT NULL,
   expiretime timestamp,
   clientip varchar(50),
   PRIMARY KEY(id)
   );
```

Appendix E. KOPEK Prototype application source code

File KOPEKPrototype.cs:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

namespace KOPEKPrototype
{
  /// <summary>
  /// Summary description for Form1.
  /// </summary>
  public class KOPEKPrototypeMainScreen : System.Windows.Forms.Form
  {
    private KOPEKFunctions accessCheckObject;
    private System.Windows.Forms.MainMenu KOPEKMainMenu;
    private System.Windows.Forms.MenuItem KOPEKModule1;
    private System.Windows.Forms.MenuItem KOPEKModule2;
    private System.Windows.Forms.MenuItem KOPEKModule3;
    private System.Windows.Forms.Button KOPEKModule;
    /// <summary>
    /// Required designer variable.
    /// </summary>
    private System.ComponentModel.Container components = null;

    public KOPEKPrototypeMainScreen()
    {
      //
      // Required for Windows Form Designer support
      //
      InitializeComponent();

      //
      // TODO: Add any constructor code after InitializeComponent call
      //
      accessCheckObject = new KOPEKFunctions();
    }

    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    protected override void Dispose( bool disposing )
    {
      if( disposing )
      {
        if (components != null)
        {
          components.Dispose();
        }
      }
      base.Dispose( disposing );
    }
    #region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
      this.KOPEKModule = new System.Windows.Forms.Button();
      this.KOPEKMainMenu = new System.Windows.Forms.MainMenu();
      this.KOPEKModule1 = new System.Windows.Forms.MenuItem();
      this.KOPEKModule2 = new System.Windows.Forms.MenuItem();
      this.KOPEKModule3 = new System.Windows.Forms.MenuItem();
      this.SuspendLayout();
      //
      // KOPEKModule
      //
      this.KOPEKModule.Enabled = false;
      this.KOPEKModule.Font = new System.Drawing.Font("Microsoft Sans Serif", 20.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
      this.KOPEKModule.Location = new System.Drawing.Point(8, 8);
      this.KOPEKModule.Name = "KOPEKModule";
      this.KOPEKModule.Size = new System.Drawing.Size(504, 104);
      this.KOPEKModule.TabIndex = 0;
      this.KOPEKModule.Text = "KOPEK INITIAL MODULE";
      this.KOPEKModule.Click += new System.EventHandler(this.KOPEKModule_Click);
      //
      // KOPEKMainMenu
      //
      this.KOPEKMainMenu.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] {

            this.KOPEKModule1,

            this.KOPEKModule2,

            this.KOPEKModule3});
      //
      // KOPEKModule1
      //
      this.KOPEKModule1.Index = 0;
      this.KOPEKModule1.Text = "Module 1";
      this.KOPEKModule1.Click += new System.EventHandler(this.KOPEKModule1_Click);
      //
      // KOPEKModule2
      //
      this.KOPEKModule2.Index = 1;
      this.KOPEKModule2.Text = "Module 2";
      this.KOPEKModule2.Click += new System.EventHandler(this.KOPEKModule2_Click);
      //
      // KOPEKModule3
      //
      this.KOPEKModule3.Index = 2;
      this.KOPEKModule3.Text = "Module 3";
      this.KOPEKModule3.Click += new System.EventHandler(this.KOPEKModule3_Click);
      //
      // KOPEKPrototypeMainScreen
      //
      this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
      this.ClientSize = new System.Drawing.Size(520, 113);
      this.Controls.Add(this.KOPEKModule);
      this.Menu = this.KOPEKMainMenu;
      this.Name = "KOPEKPrototypeMainScreen";
      this.Text = "KOPEKPrototypeMainScreen";
      this.ResumeLayout(false);

    }
    #endregion
```

```
/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]
static void Main()
{
  KOPEKFunctions accessCheckObject = new KOPEKFunctions();
  KOPEKSTARTERAXLib.KPKStarter kopekCheck;
  try
  {
    kopekCheck = new KOPEKSTARTERAXLib.KPKStarterClass();
  }
  catch
  {
    //KOPEK Client not installed, install it from local directory kclient
    System.Diagnostics.Process kopekInstallation = new System.Diagnostics.Process();
    kopekInstallation.StartInfo.FileName = "kclient\\ksetup.exe";
    kopekInstallation.StartInfo.ErrorDialog = true;
    kopekInstallation.StartInfo.CreateNoWindow = false;
    kopekInstallation.Start();
    kopekInstallation.WaitForExit();
    kopekCheck = new KOPEKSTARTERAXLib.KPKStarterClass();
  }

  if(kopekCheck.GetClientState() < 2)
  {
    kopekCheck.StartClient();
  }
  else if(kopekCheck.GetClientState() < 4)
  {
    kopekCheck.SetClientWindowForeground();
  }

  try
  {
    if(!accessCheckObject.CheckAccess("Startup"))
    {
      Application.Exit();
    }
    else
    {
      Application.Run(new KOPEKPrototypeMainScreen());
    }
  }
  catch (Exception accessException)
  {
    MessageBox.Show(accessException.Message);
  }

}

private void KOPEKModule1_Click(object sender, System.EventArgs e)
{
  try
  {
    if(!accessCheckObject.CheckAccess("Module1"))
    {
    }
    else
    {
      KOPEKModule.Text = "KOPEK Module 1\r\nClick here to return to initial module";
      KOPEKModule.Enabled = true;
    }
  }
  catch (Exception accessException)
  {
    MessageBox.Show(accessException.Message);
  }
}
```

```
    private void KOPEKModule2_Click(object sender, System.EventArgs e)
    {
      try
      {
        if(!accessCheckObject.CheckAccess("Module2"))
        {
        }
        else
        {
          KOPEKModule.Text = "KOPEK Module 2\r\nClick here to return to initial module";
          KOPEKModule.Enabled = true;
        }
      }
      catch (Exception accessException)
      {
        MessageBox.Show(accessException.Message);
      }
    }

    private void KOPEKModule3_Click(object sender, System.EventArgs e)
    {
      try
      {
        if(!accessCheckObject.CheckAccess("Module3"))
        {
        }
        else
        {
          KOPEKModule.Text = "KOPEK Module 3\r\nClick here to return to initial module";
          KOPEKModule.Enabled = true;
        }
      }
      catch (Exception accessException)
      {
        MessageBox.Show(accessException.Message);
      }
    }

    private void KOPEKModule_Click(object sender, System.EventArgs e)
    {
      KOPEKModule.Text = "KOPEK INITIAL MODULE";
      KOPEKModule.Enabled = false;
    }
  }
}
```

File KOPEKFunctions.cs:

```
using System;
using System.Security.Cryptography;
using System.Web;
using System.Net;
using System.Text;
using System.IO;
using System.Threading;

namespace KOPEKPrototype
{
  // Change these values to suit your setup.
  class Configuration
  {
    public static string GetContentServerHost ()   { return "host.domain.name"; }
    public static string GetLicenseControlScript() { return "/getlicense.php"; }
  }

  /// <summary>
  /// Access Check object. Checks access and notifies parent.
  /// </summary>
  public class KOPEKFunctions
  {
    // Generate your own 1024-bit RSA key:
    // openssl genrsa -out key.pem 1024

    // Output modulus:
    // openssl rsa -in key.pem -modulus -noout
    // Use a keyboard macro or script to convert to the below format:
    static private byte[] s_pubKeyModulus =
      {
0xC6,0xCB,0x21,0x24,0x7D,0xAB,0xDD,0x99,0x4B,0xE1,0xF2,0xAA,0x38,0x53,0x87,0x72,0x2A,
0x25,0x00,0xEF,0xCD,0xEF,0x18,0x48,0xD2,0x80,0x97,0xF1,0xFB,0xD6,0x29,0x86,0xC1,0xA6,
0x6D,0xAD,0x6F,0x27,0xE6,0xEC,0xC7,0xE1,0x11,0x7E,0x71,0x19,0x09,0xDF,0xAE,0x2D,0xC7,
0x6F,0x72,0x7C,0x1A,0x19,0x19,0x65,0x83,0x39,0x3C,0x68,0xA6,0x23,0x0F,0x94,0xE5,0x8B,
0x14,0x25,0xE8,0x76,0x12,0x3B,0x91,0xCF,0x27,0x2E,0xC4,0x7B,0x57,0x7B,0x6B,0x04,0xEB,
0x41,0x82,0xF6,0xB5,0x7D,0xD0,0xCD,0xD7,0x9D,0xE4,0x9A,0x9A,0x11,0x4C,0xF0,0x0D,0x5F,
0x36,0xC3,0xC9,0xCF,0xCB,0xF1,0x63,0x07,0xCB,0xA7,0xBD,0x3E,0x8A,0x4A,0x35,0xCF,0x8C,
0x19,0x1D,0xF3,0x88,0xA9,0x12,0x89,0x15,0x81
      };

    // Default openssl RSA exponent 65537 (0x10001)
    static private byte[] s_pubKeyExponent = { 0x01, 0x00, 0x01 };

    private RNGCryptoServiceProvider m_random;
    private RSACryptoServiceProvider m_rsa;
    public int loggedOnKOPEKUserID;

    public KOPEKFunctions()
    {
      // Setup random source
      m_random = new RNGCryptoServiceProvider();

      // Setup RSA public key
      m_rsa = new RSACryptoServiceProvider();
      RSAParameters keyInfo = new RSAParameters();
      keyInfo.Modulus   = s_pubKeyModulus;
      keyInfo.Exponent = s_pubKeyExponent;

      m_rsa.ImportParameters(keyInfo);
    }
```

```
/// <summary>
/// Verify that signature was created using key file used to
/// generate public key in this code
/// </summary>
/// <param name="dataStr">Challenge sent to server side script</param>
/// <param name="base64Signature">Response signature from server side script</param>
/// <returns>Match between challenge and signature</returns>
public bool CheckSignature (string dataStr, string base64Signature)
{
  byte[] signature = Convert.FromBase64String(base64Signature);

  UTF8Encoding ue = new UTF8Encoding();
  byte[] data = ue.GetBytes(dataStr);

  SHA1 sha = new SHA1CryptoServiceProvider();
  byte[] hash = sha.ComputeHash(data);

  RSAPKCS1SignatureDeformatter deformatter = new RSAPKCS1SignatureDeformatter(m_rsa);
  deformatter.SetHashAlgorithm("SHA1");

  return deformatter.VerifySignature(hash, signature);
}
```

```
    /// <summary>
    /// Perform HTTP request through the KOPEK payment system.
    /// This yields access control and payment if needed.
    /// </summary>
    /// <param name="uri">Path to server side license check script</param>
    /// <returns>Answer from server side script</returns>
    public string OnlineAccessCheck (string uri)
    {
      string host = Configuration.GetContentServerHost();
      string signature = null;
      string sigPrefix = "Signature: ";
      string kopekUIDPrefix = "KOPEKUID: ";
      string clientPrefix = "http://localhost:27504/KSecure____";
      string url = clientPrefix + "http://" + host + uri;

      HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);
      try
      {
        request.Timeout = 9000 * 1000; // 9000 seconds (we want ~unlimited)
        HttpWebResponse response = (HttpWebResponse)request.GetResponse();
        StreamReader sr = new StreamReader(response.GetResponseStream());
        bool signatureFound = false;
        for(;;)
        {
          string line = sr.ReadLine();
          if(line == null)
            break;
          if(line.StartsWith(kopekUIDPrefix))
            loggedOnKOPEKUserID = Convert.ToInt32(line.Substring(kopekUIDPrefix.Length));
          if(line.StartsWith(sigPrefix))
          {
            signature = line.Substring(sigPrefix.Length);
            signatureFound = true;
          }
          if(line.IndexOf("kopekid") > -1)
          {
            loggedOnKOPEKUserID = Convert.ToInt32(line.Substring(line.IndexOf("VALUE=") + 7,
line.LastIndexOf("\"")-(line.IndexOf("VALUE=") + 7)));
          }
        }
        if(!signatureFound)
        {
          response.Close();
          sr.Close();
          request.Abort();
          request = (HttpWebRequest)WebRequest.Create(url);
          request.Method = "POST";
          request.ContentType = "application/x-www-form-urlencoded";
          ASCIIEncoding encoding = new ASCIIEncoding();
          string postData = "submitted=";
          postData += HttpUtility.UrlEncode("1");
          postData += "&";
          postData += "customername=";
          postData += HttpUtility.UrlEncode("custname");
          postData += "&";
          postData += "contactname=";
          postData += HttpUtility.UrlEncode("contname");
          postData += "&";
          postData += "email=";
          postData += HttpUtility.UrlEncode("email@test.com");
          postData += "&";
          postData += "kopekid=";
          postData += HttpUtility.UrlEncode(loggedOnKOPEKUserID.ToString());
          request.ContentLength = postData.Length;
          using (Stream writeStream = request.GetRequestStream())
          {
            UTF8Encoding UTFencoding = new UTF8Encoding();
            byte[] bytes = UTFencoding.GetBytes(postData);
            writeStream.Write(bytes, 0, bytes.Length);
          }
          response = (HttpWebResponse) request.GetResponse();
          sr = new StreamReader(response.GetResponseStream());
          for(;;)
          {
            string line = sr.ReadLine();
            if(line == null)
                break;
          if(line.StartsWith(kopekUIDPrefix))
            loggedOnKOPEKUserID = Convert.ToInt32(line.Substring(kopekUIDPrefix.Length));
          if(line.StartsWith(sigPrefix))
          {
            signature = line.Substring(sigPrefix.Length);
            signatureFound = true;
          }
        }
      }
      sr.Close();
      response.Close();
    }
    catch(ThreadAbortException)
    {
      request.Abort();
    }
    return signature;
  }

  /// <summary>
  /// We append a random challenge string to each request to make it
  /// impossible to defeat the access control through replay attacks.
  /// </summary>
  /// <returns>Random challenge string</returns>
  public string GetChallengeString ()
  {
    byte[] randomBytes = new Byte[30];
    m_random.GetBytes(randomBytes);

    return Convert.ToBase64String(randomBytes);
  }

  /// <summary
  /// Check access using configuration settings
  /// </summary>
  /// <returns>Result of access check</returns>
  public bool CheckAccess(string productID)
  {
    string challenge = GetChallengeString();

    string uri = Configuration.GetLicenseControlScript() +
      "?productid=" + HttpUtility.UrlEncode(productID) +
      "&challenge=" + HttpUtility.UrlEncode(challenge);
    string signature = OnlineAccessCheck(uri);
    bool accepted = CheckSignature(HttpUtility.UrlEncode(challenge), signature);

    if(!accepted)
    {
    }
    return accepted;
  }

 }
}
```