Dinkey Pro and Dinkey FD

Developer's Guide

Microcosm Ltd

Southfield House
2, Southfield Road
Westbury-on-Trym
Bristol
BS9 3BH
United Kingdom

E-mail: dinkey@microcosm.com
Web: www.microcosm.com

Manual printed March 2024 for Dinkey Pro SDK version 8.0.0

Contents

Introduction

Overview

A Dinkey dongle is a small piece of hardware which is part of a sophisticated system designed to protect your software from illegal copying. You can lock your software to a dongle so that each time you run your protected program, it will check for the presence of the dongle. If the right dongle is found, the program will be executed, otherwise it will not run.

There are 2 types of dongle: Dinkey Pro and Dinkey FD. Dinkey Pro is a very secure driverless dongle. Dinkey FD is a flash disk that also can be used as a dongle. The Pro and FD dongles have similar functionality so it is easy to switch from using one to the other.

For each type of dongle there are 3 models: Lite, Plus and Net.

The **Lite** model allows you to protect your software simply and quickly without the need to program dongles. Note that the security of the Dinkey FD Lite is not as advanced as the other models.

The **Plus** model not only allows you to protect your software, but also to control it and store your own secure data. You can set an expiry date, use an execution counter or control the use of various features in your software. You also have the facility to alter these protection parameters remotely. So, for example, at a later date you could change the expiry date on your program just by issuing a secure update code by e-mail or phone.

The **Net** model has all the features of the Plus model but allows you to use one dongle per network instead of one dongle per machine. In addition, you can limit the number of simultaneous network users of your program.

The SDK that is supplied with the Dinkey system is comprised of the following:

DinkeyAdd - a program that adds protection to your software and programs the dongle if necessary.

DinkeyRemote - a program that generates update codes for the remote changing of protection parameters.

DinkeyChange - a program that accepts the update code that DinkeyRemote generates and modifies the dongle accordingly.

DinkeyLook - a program that searches all the Dinkey dongles attached to a computer and displays various parameters stored in those dongles.

DinkeyServer - a program (or service or daemon) that needs to be run for the Net dongle to function correctly.

DinkeyManual - a PDF manual showing how to use the Dinkey dongle system. A Windows help file is also provided.

Modules folder - various runtime modules that contain code to communicate with our dongles.

Samples folder - sample source code to protect your software using the API Method of protection. We strongly recommend that you read the "readme" file in the sample code directory for the programming language that you are using for important information on integrating Dinkey Pro with your programming language.

SDSN - the Software Developer's Serial Number (SDSN) is a number that is uniquely allocated to your company when you order the Dinkey Pro SDK. The SDSN is burnt into each dongle we send to you. It is also used to encrypt the SDK software so that your SDK is unique to your company and will only be able to access dongles belonging to you. (The demo SDK has an SDSN of 10101).

For each SDSN we have registered postal and e-mail addresses. We only send dongles to the registered postal address and only send SDK updates to the registered e-mail address for that SDSN. This means that other companies cannot order dongles with your SDSN or view their contents.

Product Code - a string that identifies one of your products from another. This makes it possible to have dongles that protect one product and other dongles that protect another product. For Lite dongle models, which cannot be programmed, you need to specify the Product Code when ordering. For other dongle models you can use DinkeyAdd to program the dongle with the Product Code of your choice. Note that the Product Code does not need to be kept secure - it is just used to identify your dongle with one of your products.

Dongle Number - a 32-bit integer which is unique to each dongle. (It is possible to order your own dongle numbers – in which case they cease to become to unique. Only in very rare cases is this useful, such as if you want to identify certain customers with particular serial numbers).

A unique dongle number means that an update code generated by DinkeyRemote will only work for a particular dongle. It can also be useful for identifying a particular dongle. However, we do not recommend that you make the dongle number a part of your protection strategy because it would mean protecting your software differently for each dongle. Normally you would protect your software to an SDSN and Product Code because then it can be used with any dongles with these values that you send to your customers.

How It Works - the Dinkey Pro system protects your software by locking it to a dongle. In particular, your software is locked to a dongle that contains your SDSN and the Product Code you specified when you protected it using DinkeyAdd. In addition, you can protect your software to specified dongle models or even to a particular dongle number, although this is very rarely needed.

You can use either the **API Method** (calling the Dinkey Pro API from your source code) or the **Shell Method** (automatically adding protection and encrypting your program code and data) to protect your software.

You can use a **Temporary Software Key** to temporary replace a dongle if it is damaged, so a customer can continue using their protected software until a replacement dongle arrives. A **Demo Software Key** allows you to securely produce fully functional time-limited demos of your software. These are both temporary solutions which are purely software-based and require an internet connection.

If you want to manage all your dongle updates using an online database then **DinkeyOMS** is the answer. This system will simplify the update process and can even automate it if used in conjunction with an online payment system. In addition, DinkeyOMS can be used to block dongles that a user has reported as lost, damaged or stolen.

System Requirements

Dinkey Pro and FD are compatible with all client versions of Windows from Windows XP onwards; all Windows Server versions from 2003 onwards; Linux distributions using kernel version 2.6.18 or higher and Mac OS X 10.10 Yosemite or newer (Intel and Silicon chipsets). Linux is supported on x86, x64 and ARM (both aarch32 and aarch64).

Dinkey Pro dongles do not require drivers on any of these platforms.

Dinkey Pro Features List

1) No need to install drivers (it uses drivers built into the operating system).

2) The Dinkey Pro hardware is based on smart card technology. This is proven to be very secure – it is widely used in the security industry.

3) All communication with the Dinkey Pro dongle uses very strong encryption techniques – a mixture of RSA encryption and strong 128-bit encryption. Using RSA encryption means that even if an attacker discovered the public key our software then they could not work out the private key in the dongle. This means it is not possible to emulate the hardware dongle.

4) Many techniques to improve the security between your code and our code. This is lacking in many dongle systems but it is often the weakest part of the protection system.

5) The Plus and Net models have an 8K secure data area. You can protect up to 255 programs or modules with independent parameters and specify up to 20 algorithms.

6) The Plus and Net models can be used to encrypt or decrypt data you pass to it. All encryption is executed in the dongle itself.

7) Strong anti-debug code – a hallmark of all our software protection systems.

Dinkey FD Features List

1) No need to install drivers (it uses drivers built into the operating system).

2) As well as having the functionality of a security dongle it can also be used as a flash disk. You can distribute your software on the dongle itself!

3) Many techniques to improve the security between your code and our code. This is lacking in many dongle systems but it is often the weakest part of the protection system.

4) Strong anti-debug code – a hallmark of all our software protection systems.

5) The Dinkey FD Plus and Dinkey FD Net have all the features and security of the Dinkey Pro dongle.

6) The Dinkey FD Lite has very fast access times and the protection check is very quick.

How This Manual is Organised

The following is a summary of the next few chapters and how you can use them to help you protect your software.

The Quick Tour leads you through the process of protecting a simple program and modifying some of the protection parameters. It is recommended that you read this chapter as it will make you familiar with the Dinkey software and how it operates. If you feel confident you can omit this chapter and proceed to the next chapters that describe each step in more detail. Please note that reading this chapter is by no means a replacement for reading the rest of the manual.

You need to read the chapters on Protection Method, Modifying your Code (if applicable) and Adding Protection (DinkeyAdd) to successfully understand how to protect your software.

The chapter on Remote Changing of Parameters describes how you can modify data in the dongle remotely. It only applies to the Plus and Net models.

Please remember that the security level of your protected application depends on how you implement the Dinkey dongle system. To achieve the best levels of security you should not just copy the sample code but fully understand how the product works and adapt the sample code using some of the methods suggested in the Increasing Your Protection chapter.

Understanding Software Protection

Introduction

In order for your software to have the maximum protection available it is important to understand the principals behind software protection and how it can be beaten.

In this chapter we outline the different techniques an attacker can use to try to beat software protection systems and also methods you can employ to combat these attacks.

Dongle Emulation

The attacker monitors communication between our software and the security dongle. If they can understand the communication then they can write a program that mimics the output from the dongle and therefore use your software without the need for a dongle.

The Dinkey Pro and FD dongles beat this attack on many levels. Firstly, we use random elements in our communication with the dongle so simple "record & playback" techniques do not work.

Secondly, our own software is heavily encrypted on many levels and uses strong anti-debug techniques so that it is extremely difficult for an attacker to understand what our code is doing.

Finally, we use RSA (Public Key / Private Key encryption) when communicating with our dongles. Not only is this a very strong level of encryption but even if the attacker could extract the public key from our software (and we don't think they could!) then they would not be able to work out the private key stored in the dongle. Without this they cannot emulate the dongle.

In conclusion, we are extremely confident that no attacker can produce a solution that will emulate our dongle system.

Replacing your Protected DLL / Plugin

If you are using one of our object modules or are using the Shell method of protection then this case does not apply. However, all languages except for C/C++ and Delphi must call a DLL or special Plugin to implement the API method of protection.

In this technique the attacker tries to understand the communication between your software and our DLL or Plugin. If the attacker can understand this communication then they can replace the DLL or Plugin and remove the necessity of the dongle. This is the easiest method for an attacker to use and therefore by far the most common approach.

Furthermore it demonstrates that no matter how advanced the dongle is itself in terms of encryption, if the user implements the dongle system in a weak way (so that the communication between their program and the DLL can be easily understood) then it can still be easily beaten. A number of our competitor's dongle systems, unfortunately, can be beaten in this way.

We provide many techniques to combat this approach. All communication using our API calls can be encrypted in a way that differs for every call. We provide a technique to check that the program calling your DLL is actually your program (and not an attacker trying to glean extra information). As far as we know we are the only company that provides either of these techniques. Other techniques are discussed in the section below.

Software Patching

In this technique the attacker tries to disassemble or debug your code and remove the API calls that check for the dongle. They will also need to set the return values or data modified by the API to the values expected by your code.

This requires some skill and patience on the part of the attacker – but how much patience depends on how you call the API! If you just make one API call and only check to see whether the result is true or false then you are making it much easier for the attacker to use this technique. If, however, you check many of the values that are returned from our API, in different places in your code then you are making it much more difficult for them (especially if you encrypt your API communication).

Rather than comparing a returned value with the expected result, it is more secure to use the returned value in your code as if it was the expected value. If it is the correct value then your program will continue in the correct manner. However, if it is the wrong value then your program will fail in some way. This makes it much more difficult for the attacker because they have to understand how your program works in order to patch the correct value.

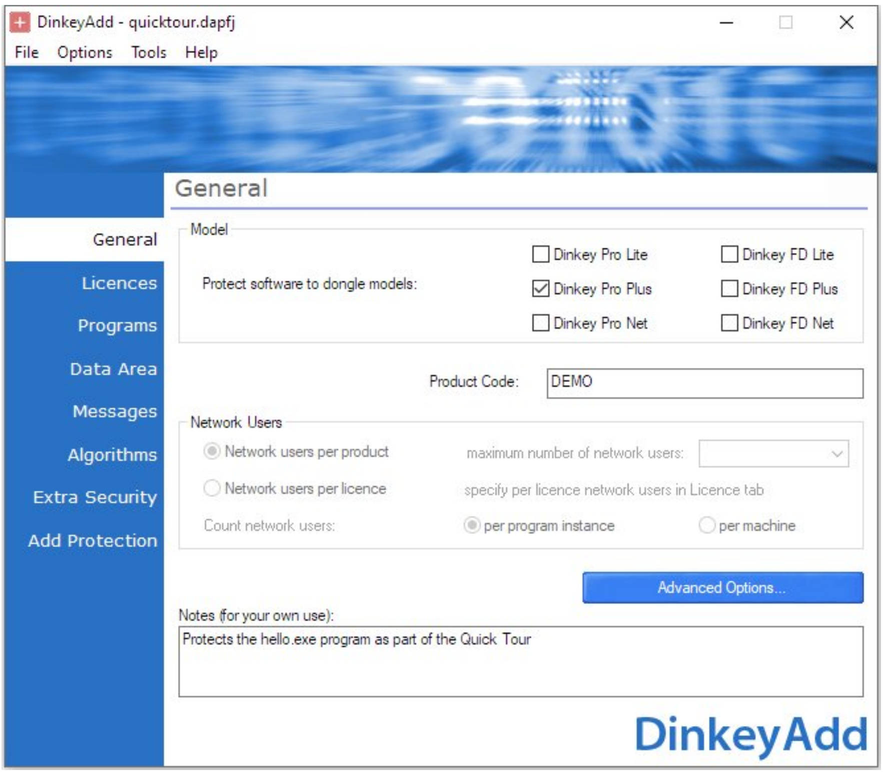
Another strong technique you can use to make this attack very difficult is to use one or more algorithms in the dongle. Further techniques are discussed in the section Increasing Your Protection.

Quick Tour

Protecting a Sample Program

This section will guide you through the process of protecting a simple program under Windows. If you want to protect a Linux or Mac program then please install the SDK for the relevant platform and follow the instructions in the Quick Tour directory instead.

We will choose the *hello* program located in the QuickTour folder of the installation directory. To protect this file you must run DinkeyAdd. Do this now and open the *quicktour.dapfj* settings file which is found in the QuickTour folder (in the same folder where you installed Samples). Your screen should look something like this:



Model

Protect software to dongle models: ☐ Dinkey Pro Lite ☐ Dinkey FD Lite ☑ Dinkey Pro Plus ☐ Dinkey FD Plus ☐ Dinkey Pro Net ☐ Dinkey FD Net

Product Code: DEMO

Network Users

● Network users per product maximum number of network users:

○ Network users per licence specify per licence network users in Licence tab

Count network users: ● per program instance ○ per machine

Notes (for your own use):

Protects the hello.exe program as part of the Quick Tour

DinkeyAdd with quicktour.dapfj loaded

Please modify the Model settings so that only the model of dongle you are using is selected. If you are using a Lite dongle then you will also need to modify the Product Code to match the Product Code in the dongle. If you are unsure of the dongle model or Product Code then attach the dongle and then run DinkeyLook.

Now choose the **Programs** tab and click on the **Modify Selected** button. You need to modify the Input Pathname to be the full path and name of the file to protect (hello.exe in the QuickTour folder). You also need to modify the Output Pathname. The protected program will be written to this file and the original unprotected program will remain untouched (so long as the Input filename is different to the Output filename).

You are now ready to protect hello. Attach the dongle to your computer and then select the **Add Protection** tab and click on **Add Protection Now**. This will program the dongle (not necessary for Lite models) and protect the hello program so that it will only run if the dongle is present.

Try that now. With the dongle attached the protected hello program should run normally. If you remove the dongle from your computer and try running the protected hello program it will display an error message and then terminate.

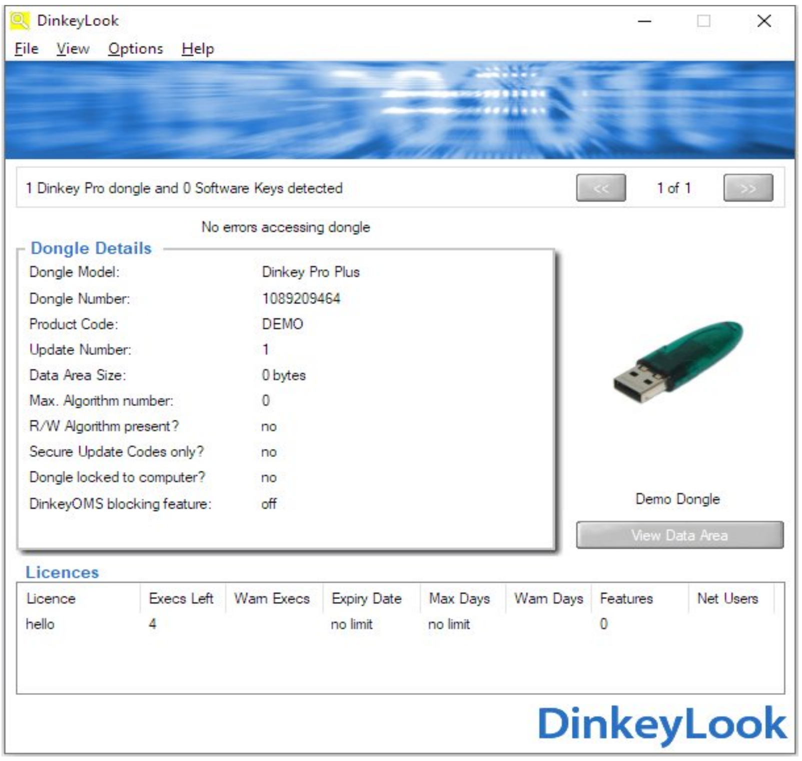
Note – if you are using a Net dongle then there is an extra step involved. You need to run *DinkeyServer* on the computer that has the dongle attached. For more information see the Network dongles chapter.

You have added protection to a simple program. To find out how to protect *your* program you need to read Protection Method. If you have a Plus or Net model then you may like to read the next section which shows you how to modify the protection parameters stored in the dongle.

Modifying Protection Parameters

If the hello program was protected to a Plus or Net dongle then it was initially limited to 5 executions. To find out how many executions you have left please attach the dongle to your machine again and run DinkeyLook.

It should display a screen similar to this:



1 Dinkey Pro dongle and 0 Software Keys detected 1 of 1

No errors accessing dongle

Dongle Details

Dongle Model:	Dinkey Pro Plus
Dongle Number:	1089209464
Product Code:	DEMO
Update Number:	1
Data Area Size:	0 bytes
Max. Algorithm number:	0
R/W Algorithm present?	no
Secure Update Codes only?	no
Dongle locked to computer?	no
DinkeyOMS blocking feature:	off

Demo Dongle

Licences

Licence	Execs Left	Warn Execs	Expiry Date	Max Days	Warn Days	Features	Net Users
hello	4		no limit	no limit		0	

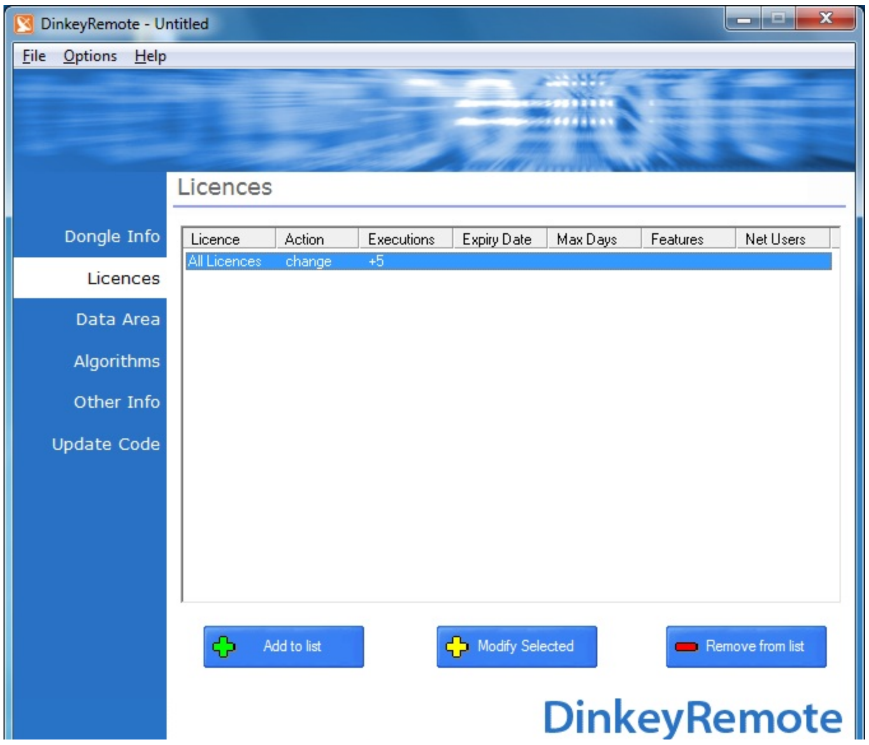
DinkeyLook displaying the contents of the protected dongle.

The value under *Execs Left* tells you how many more times you can successfully run the hello program. Please repeatedly run the hello program until it comes up with the following message: *MyProgram has used up all of its allowed executions*. You have now successfully run hello your allotted five times and you cannot run it again. The value of Execs Left in the dongle is 0. However, you can change this value by running the DinkeyRemote and DinkeyChange programs.

Now run DinkeyRemote. You first must enter the Product Code, Dongle Number and Update Number. Although you can find this information from DinkeyLook it is also displayed by DinkeyChange so please run this program now. You can copy the information using copy and paste.

Once you entered these values please click on the Licences tab and then on the *Add to List* button. Enter the value 5 into the Executions box, select *Add* and then click OK. This will change all licences in the dongle to have 5 more executions.

Your screen should look similar to the one below:



Licences

Licence	Action	Executions	Expiry Date	Max Days	Features	Net Users
All Licences	change	+5				

DinkeyRemote showing the Licences tab

Now choose the **Update Code** tab. Normally you will probably produce strongly encrypted code but for this demonstration please select *Generate Short Update Code* and click the **Generate Update Code** button. By default this update code is copied to the clipboard and also to the file specified in this tab. Now paste this code into DinkeyChange and click on **Make Changes to Dongle**. If everything has been done correctly you will see the message *the dongle update has been completed successfully*.

The executions value has now been restored to 5. You can run DinkeyLook to prove this (or if DinkeyLook is already running you can refresh).

Now you have to learn how to use DinkeyAdd, DinkeyLook, DinkeyRemote and DinkeyChange to protect your software and modify the protection parameters in the dongle. To understand how to use these programs in more detail please read the remaining chapters before you protect your own software.

Protection Method

Overview

There are two methods of protecting your software:

- The **Shell Method** – automatically adds protection to Windows programs and DLLs, and Linux binaries and shared libraries. It also encrypts your software preventing de-compilation of your code. It is quick and easy to implement.
- The **API Method** - allows you to call our API from within your own code. It gives you the greatest flexibility over when and where you communicate with the dongle. It also allows you to control which parts of your code the customer can use. It takes more time to implement this method because you need to understand our API but the resulting protection can be made very strong.

Of course, if you want to have the maximum security available then you can use both methods of protection.

With some languages, such as Java and Python, you have to use the API Method because they do not output native executable code (see the readme file in the sample code for your programming language for more details).

The Shell Method

The Dinkey Pro SDK can directly protect both 32-bit and 64-bit Windows and Linux executable programs and DLLs (shared libraries) by putting a protective "shell" around them. It can also protect PDF files (the output is a Windows-only executable file that launches a viewer that displays the PDF file).

The Shell Method requires no preparation - just run DinkeyAdd to protect your program software. If your software is a .NET program then the Shell Method will produce one or two DLLs that need to be distributed with your protected program.

One big advantage of the Shell Method is that it encrypts parts of your code and data (the API method cannot do this). It also adds extra layers of anti-debug code. However, you do not have the flexibility and control that you get when you use the API method.

If your Shell-protected executable accesses external data files then you can protect (encrypt) these data files by using the Data File Encryption Tool in DinkeyAdd.

In addition to the protection offered by the Shell Method there are two options to increase the security still further: **Extra Anti-Debug** and **Extra Anti-Piracy**. Both of these options can affect the speed of your protected program. You can choose the strength of the extra anti-piracy level to accommodate this, or even turn off one or both options if necessity demands.

Notes:

- If you Shell-protect a DLL (or shared library under Linux) then if the dongle is not detected at runtime the DLL will not load correctly (under Linux it will generate real time signal sigrtmin+22). Also, if the background protection check fails then the entire process will be terminated (after the Background Check Error message is displayed). For a .NET DLL, on error the entire process is terminated.
- Both types of Software Key are not compatible with a Shell-protected standard Windows DLL (but will work with .NET and .NET Core Shell-protected DLLs).
- The Shell Method is not supported under macOS.
- The Shell Method can only protect Universal Windows Platform apps (previously called Windows Store apps or Metro-style apps) to Net dongles.

- Some Windows executable files have data that is attached to the end of file but is not part of the executable file format. In this case the Shell Method is not able to encrypt this data (e.g. flash swf files converted to an executable file).
- Data File Encryption and Extra Anti-Piracy are not supported under Linux.
- The Extra Anti-Piracy setting has no effect for the .NET Shell Method as this technique is an integral part of the .NET Shell protection and hence is always enabled.

Read the section Adding Protection to proceed using the Shell method. We also advise you to look at the Readme file in the sample code directory for your programming language as it can sometimes contain information about the Shell Method too.

When the Shell is applied to .NET programs there are additional considerations to take in account. These are listed in the section below.

Shell Method for .NET Assemblies

Requirements:

- It can protect assemblies that target all .NET frameworks from 2.0 onwards on Windows.
- .NET Core is supported from version 3.0 onwards on Windows and Linux (x64 chipset).
- The assembly must be pure MSIL. Mixed assemblies are not supported.
- Universal Windows Platform apps are not supported by the .NET Shell Method.

Unlike native executables, .NET programs are compiled to Intermediate Language (IL) which means that they are more easily reverse engineered. The Shell Method for .NET programs locates each method and function in your program and removes the IL code. This code is then stored encrypted and at runtime is executed dynamically. This is far more secure than code obfuscation techniques employed in other protection systems.

If you are protecting a .NET DLL note that the protection check will happen the first time you call a method/function in the DLL, rather than when the DLL is loaded (.NET DLLs do not have a main entry point). You may wish to design your code so that the protection check does not occur at an inappropriate time.

There is a small amount of performance overhead associated with the .NET Shell. You will not normally be affected by this and can apply .NET Shell in its default mode of operation (protecting as much of your program as possible!) without needing to change anything, but there may be situations when you need to tailor the protection in which case please read the next section.

Advanced Use of .NET Shell Method

There may be situations when you want to control exactly what parts of your program are protected or excluded from protection. For example, there might be a lot of code in your program that you are not interested in protecting (e.g. GUI code) and some parts which you definitely do want protected (e.g. important algorithms). Or maybe applying the Shell Protection slows down a particular function in your code. In some cases you have to exclude code because it interferes with the Shell Method, such as methods that rely on Reflection (e.g. using the *DeclaringType* property of the *MethodBase* class).

You can define these requirements by applying the System.Reflection.ObfuscationAttribute attribute to methods and classes in your code, setting the Feature property as follows:

Feature="Dinkey:OptIn" to include the class/method for protection.

Feature="Dinkey:OptOut" to exclude the class/method from protection.

Only one form of the attribute is to be used in an assembly at once (either OptIn, or OptOut, but not both). If you use OptOut attributes, then Shell protection will be applied to all of your code except those parts explicitly marked OptOut. Conversely, if you use OptIn attributes, then Shell protection will only be applied to those parts of your code marked OptIn.

The ObfuscationAttribute is inherited so when applied to a class it therefore applies to all methods and nested classes within that class. An explicitly defined ObfuscationAttribute will take precedence over an implicit (inherited) one.

Example 1:

```
class C
   {
       public int F1(int i)
       {
           // ...
       }

       public int F2(int i)
       {
           // ...
       }
   }
   // Results: F1() and F2() both protected
```

Example 2:

```
[System.Reflection.ObfuscationAttribute(Feature="Dinkey:OptIn")]
class C1
{
    public int F1(int i)
    {
        // ...
    }

    public int F2(int i)
    {
        // ...
    }
}

class C2
{
    public int F1(int i)
    {
        // ...
    }

    public int F2(int i)
    {
        // ...
    }
}
// Results: C1::F1() and C1::F2() both protected. C2::F1() and C2::F2() not protected.
```

Example 3:

```
class C
{
    public int F1(int i)
    {
        // ...
    }

    [System.Reflection.ObfuscationAttribute(Feature="Dinkey:OptOut")]
    public int F2(int i)
    {
        // ...
    }

    public int F3(int i)
    {
        // ...
    }
}
// Results: F1() and F3() both protected, F2() not protected.
```

The API Method

This method allows you greater flexibility. You can control exactly when and how often the dongle is checked and take whatever action you want should the dongle not be present. However, if you use this method you have to make some changes to your source code. You will also need to fully understand our API because if you implement this method weakly then your software will have limited protection. Conversely, if you follow all the guidelines outlined in this manual then your protection will be very strong.

There are three stages to this method:

1. Modify your code so that it checks for the presence of a Dinkey dongle. You should call the API in different parts of your code. It is advisable to implement some of the techniques discussed in Increasing Your Protection.
2. Link the appropriate module for the language your program is written in. This may be a static object module or a dynamic library (.dll for Windows, .so for Linux or .dylib for Mac) or a Plugin. For a complete list of all the modules available see Protection Modules.
3. Run DinkeyAdd to lock your software. If you have linked a static library then you need to protect your program but otherwise you protect our runtime module (e.g. dpwin32.dll) rather than your program itself. Because your program is linked to this DLL/Plugin then it is also protected.

Read the sections “Modifying Your Code” on page 13 and “Adding Protection (DinkeyAdd)” on page 19 to proceed using the API method.

Modifying Your Code

Overview

[API Method only]

If you are using the API Method you need to make some modifications to your code in order to call the Dinkey Dongle API that interrogates the dongle.

Information is exchanged between your software and our module via a structure that we call the **DRIS** (Dinkey Runtime Information Structure). How this structure is filled-in by you determines what it does. See “Structures” for a complete description of the DRIS structure. We will call each member of the DRIS structure a “field”.

There is only one API call for doing this, DDProtCheck. It takes 2 parameters and returns an integer:

Integer **DDProtCheck** (DRIS structure, Data)

The Data parameter is used to write data to the dongle data area or read data from the dongle data area. This parameter is used in those languages that cannot implement pointers (in a structure). For languages that can support pointers this data is written/read using the rw_data_ptr field of the DRIS and the Data parameter is set to NULL. The sample code demonstrates which method you should use for your programming language.

You should fill-in the relevant fields in the DRIS before each call to DDProtCheck. These are the fields that must be filled-in each time:

header	Must be set to “DRIS”.
size	Set to the size of the DRIS structure.
function	Tells our API what you want it to do.
flags	Enable the function to carry out other (optional) tasks. You can specify more than one flag by OR-ing them together. You can also set it to 0 if you want no flags to be set.

The following fields are always filled in by the API:

ret_code	return code from the function (0 = success)
ext_err	extended error which may tell us more error information.

Which other fields need to be filled-in or which fields will return with valid information depends on the function and flags values that are set. Please find a list of valid functions and flags below. For each function and flag there is a key indicating which dongle model it applies to (L = Lite, P = Plus, N = Net).

If you try to use a function that is not valid for the dongle detected then an error will occur. If you use inappropriate flags for the dongle detected (e.g. start a network user for a Lite dongle) then these flags are ignored.

*Note – for each call using the WRITE_DATA_AREA, READ_DATA_AREA, ENCRYPT_USER_DATA or DECRYPT_USER_DATA you are limited to a length 1024 bytes (1K) of data. This is limited for speed reasons. If you need to use more data then make multiple calls to these functions.

DDProtCheck is thread-safe.

Functions

PROTECTION_CHECK (1) L P N

Description: Searches for a dongle that meets the criteria specified in DinkeyAdd. Returns various parameters stored in the dongle.

Inputs: the compulsory fields listed above.

Outputs: all fields (relevant to the dongle model) except for alg_answer.

EXECUTE_ALGORITHM (2) L P N

Description: As for PROTECTION_CHECK but also executes one of the algorithms you specified in DinkeyAdd. If you are using a Lite dongle then it will execute the algorithm contained in the dongle.

Inputs: the compulsory fields plus:

alg_number	the number of the algorithm to execute. If you are using a Dinkey Lite dongle then this field is ignored.
var_a…var_h	the variables for the algorithm to use.

Outputs: all fields (relevant to the dongle model). The answer to the algorithm calculation is returned in the alg_answer field.

WRITE_DATA_AREA (3) P N

Description: As for PROTECTION_CHECK but also writes the data you specify to the data area of the dongle at the offset you specify.

Inputs: the compulsory fields plus:

rw_offset	the offset in the dongle data area to write to.
rw_length	the length of the data to write (this is limited to 1K*).

The data is passed in either rw_data_ptr or the Data argument of DDProtCheck depending on the flags set.

Outputs: all fields (relevant to the dongle model), except for alg_answer.

READ_DATA_AREA (4) P N

Description: As for PROTECTION_CHECK but also reads the specified length of data from the specified offset in the dongle data area.

Inputs: the compulsory fields plus:

rw_offset	the offset in the dongle data area to read from.
rw_length	the length of the data to read (this is limited to 1K*).

The data is passed in either rw_data_ptr or the Data argument of DDProtCheck depending on the flags set.

Outputs: all fields (depending on dongle model), except for alg_answer.

ENCRYPT_USER_DATA (5) P N

Description: As for PROTECTION_CHECK but also encrypts the data you pass. The dongle will encrypt the data using one of 3 encryption keys that are programmed using DinkeyAdd. These keys are based on your SDSN and Product Code and so will be the same for each dongle associated with a particular product. We recommend that you pad the data you want to encrypt so that it is a multiple of 8 bytes in length.

Inputs: the compulsory fields plus:

rw_length	the length of the data to encrypt (limited to 1K*)

data_crypt_key_num	the number of the encryption key to use (1-3).

The data is passed in either rw_data_ptr or the Data argument of DDProtCheck depending on the flags set.

Outputs: all fields (depending on dongle model), except for alg_answer.

DECRYPT_USER_DATA (6) P N

Description: As for PROTECTION_CHECK but also decrypts the data you pass.

Inputs: the compulsory fields plus:

rw_length	the length of the data to decrypt (limited to 1K*)
data_crypt_key_num	the number of the encryption key to use (1-3).

The data is passed in either rw_data_ptr or the Data argument of DDProtCheck depending on the flags set.

Outputs: all fields (depending on dongle model), except for alg_answer.

FAST_PRESENCE_CHECK (7) L P N

Description: Checks for the presence of a valid dongle very quickly but does not perform any strong security checks. No flags are valid with this function. You can use this function to very quickly detect whether the dongle is present. It is useful if you need to check for the presence of the dongle during some very processor-intensive code. In general this call is NOT recommended as it has reduced security (use PROTECTION_CHECK instead).

Inputs: the compulsory fields

Outputs: Only ret_code, ext_err, type, model, sdsn, prodcode, dongle_number. Other fields are not valid.

STOP_NET_USER (8) N

Description: terminates the network user belonging to this program (or the one specified in alt_licence_name if you have specified USE_ALT_LICENCE_NAME). Note – when a program terminates it will automatically terminate the network user associated with it, so this function is *not* normally needed. If you use this function then a protection check will *not* be performed.

Inputs: the compulsory fields

Outputs: Only ret_code and ext_err. Other fields are not valid.

Flags

DEC_ONE_EXEC (1) P N

Description: decrements the execution counter value by 1.

DEC_MANY_EXECS (2) P N

Description: decrements the execution counter value by the value specified in the execs_decrement field.

START_NET_USER (4) N

Description: starts a new network user for this process.

USE_FUNCTION_ARGUMENT (16) P N

Description: Tells our software that you want to use the Data parameter instead of the rw_data_ptr field in the DRIS.

CHECK_LOCAL_FIRST **(32)** N

Description: Tells our software to always look in the local machine for a dongle before looking across the network (default is to look first where we last found a dongle). You would only consider using this flag if you locked your software to both a local dongle and a network dongle using DinkeyAdd.

CHECK_NETWORK_FIRST **(64)** N

Description: Tells our software to always look across the network for a dongle before looking at the local machine (default is to look first where we last found a dongle). You would only consider using this flag if you locked your software to both a local dongle and a network dongle using DinkeyAdd.

USE_ALT_LICENCE_NAME **(128)** P N

Description: Perform the protection check but instead of using the licence connected to the program use the one specified in the alt_licence_name field. The values returned in the DRIS will be for alt_licence_name and not for the licence connected with the program you are calling from. Also network users will be started and stopped for alt_licence_name rather than the usual licence. Note that the licence name is case-sensitive.

DONT_SET_MAXDAYS_EXPIRY **(256)** P N

Description: Normally if the 'max days' field is set, the first time a customer runs your software then the expiry date is calculated by adding the max days value to the current date. If you set this flag this will not happen. This can be useful if you want to return some information to the user before they have properly started using your software.

MATCH_DONGLE_NUMBER **(512)** L P N

Description: Normally during a protection check our software will find the first dongle that matches your SDSN and Product Code. However, if you specify this flag (and set the dongle number field of the DRIS) then you can restrict the search to the dongle number you specified. This can be useful if you want to view details from more than one dongle. First, you can enumerate the dongle numbers for a given Product Code using the DCGetInfo function in the DinkeyChange module and then call DDProtCheck specifying this flag for each dongle number in turn.

DONT_RETURN_FD_DRIVE **(1024)** L P N

Description: If a Dinkey FD dongle is detected then do not return the drive letter (mount name) of the flash disk part of the dongle in the DRIS. Many virtual machine environments do not preserve the USB device hierarchy correctly and so in this case our code cannot detect the driver letter (mount name) causing a protection check to fail. It is recommended to set this flag if you are using a Dinkey FD dongle and you do not need to know the drive (mount name) of the flash disk.

How to Use the API

Fields that are not used as input or output in a particular implementation should be set to random values. This makes it more difficult for attackers to understand what is happening during the API calls.

You must check the return value from DDProtCheck and then decide on what action to take yourself. A non-zero return code indicates failure and in this case it is recommended that you display the return code along with the extended error code.

You can also check the other fields in the DRIS to see if they have the correct values. Whilst this is not necessary, it is recommended since it improves the security of the protection. It is also advised that you check these fields in another part of the program, and preferably in different places to make it more difficult for any potential attackers.

The details for calling DDProtCheck vary according to the language your code is written in. We supply full working sample code in many different languages in the **Samples** folder. **We strongly recommend that you**

look at the sample for your programming language and read the appropriate readme.txt file which contains essential information.

Please note that the sample code is a guide so that you can understand how to implement our software. Rather than just blindly copying the sample code it is best to choose the functions that you want and to modify your code using some of the ideas suggested in the chapter "Increasing your protection".

If you are using our DLL module then in order to remove potential confusion between your copy of the DLL and other products' copies of the DLL, we recommend that you rename it to a name of your choosing. (This is difficult if you are using Microsoft Visual C++ and in this case you are not recommended to rename the DLL. Please look at the sample code in the Samples\C directory for more information on this).

You can also remove confusion by installing the DLL to the installation directory rather than the Windows System directory.

Debug Modules

For some languages our anti-debugging code is so strong that if you try to debug or run your program in the development environment it may cause the debugger to crash or not function properly. For example, it will cause the debugger to crash when using Visual C++, Delphi, C++ Builder, or .NET languages (running 64-bit code).

If you want to debug your code then you can remove the protection check code and do this. Many Developer Environments have a debug build so that you can conditionally compile your code in debug build to remove the protection checks.

Another option is to use the special "debug" modules that we provide. Then you can call the protection check as normal and still debug your code. There are a number of restrictions when using the debug modules:

1) It will not correctly start a network user.
2) It does not support encrypting the DRIS.
3) The DDGetNetUserList will return error 428 (not implemented).
4) The protection check will take slightly longer than normal.
5) A temporary subprocess is created which may be (falsely) detected as malicious by anti-virus programs. If this is a problem you can whitelist the directory of the subprocess. Note - the subprocess is created in the %temp%\microcosm directory, or in the directory specified in the DD_DEBUG_TEMP environment variable, if specified.

These modules are not intended for release so you should only use them in the debug build of your project. Use the standard module for the release build of your code.

Encrypting API Parameters

To further increase security it is possible to encrypt both the DRIS and Data parameters used in the DDProtCheck call. This makes it very difficult for an attacker to understand what parameters you are passing to our software and what parameters our code is returning back to your software.

If you select to encrypt the DRIS in the Extra Security section of DinkeyAdd then our API calls will expect the DRIS to be encrypted. DinkeyAdd will generate sample code to do this based on the 3 encryption parameters that you specify in DinkeyAdd. Also, at runtime you should generate 2 random numbers that will act as encryption seeds for this encryption. You pass them in the seed1 and seed2 fields of the DRIS. This makes the encryption different for every call.

You can also encrypt the data that is passed when you want to read and write to the dongle data area or use the dongle to encrypt and decrypt data. In this case you select "Encrypt data passed to API" in DinkeyAdd. In this case you can either specify 3 encryption parameters or for extra security use the r/w algorithm. If you use the r/w algorithm the encryption parameters are extracted from the answer to this algorithm using the variable values var_a to var_h that you specify in the DRIS. The r/w algorithm has a restricted set of operations so that you can specify random values for the variables and still end up with a sensible answer.

Please do not get confused between using the dongle to encrypt data and encrypting data yourself that you pass to our API. When you encrypt data to send to our API it is just to make it more difficult for an attacker to intercept the data that you are passing to us.

You need to call CryptDRIS both before and after calling DDProtCheck. You should call CryptApiData before writing or encrypting data and after reading or decrypting data. We recommend that you look at the sample code we provide to see how encrypting the DRIS and Data parameters is implemented in your language. We also recommend that the calls to encrypt the DRIS and Data are separated from the call to the DDProtCheck function.

Protection Check Duration

There are many factors that can affect the speed of a protection check.

Typically a standard API protection check will take 100-200ms. For a Net dongle you have to add the time taken for network communication – this can vary from network to network. Other factors include:

- Decrementing executions and executing algorithms will add a small amount of time.
- Encrypting the DRIS and Data passed to our API will take about 30% longer.
- Reading and writing data to the dongle data area will take longer depending on how much you read and write. For example writing 1024 bytes (the maximum) can take around 1.4s and reading 1024 bytes about 1.2s.
- Under Windows a protection check is about twice as slow running 32-bit code compared to 64-bit code.
- If you use the Shell Method in addition to the API Method of protection then the extra anti-debug option (Windows only) can slow down and API protection check considerably – it will be about 300% slower. In this case you may want to consider turning this option off, especially if you read and write large amounts of data.

Please note these timings are purely a guide and can vary from machine to machine and Operating System. If many of these factors apply then the slowing effects are multiplied.

Note – while we do recommend you check the dongle throughout your code you should **not** check the dongle constantly (e.g. once per second) because a protection check could take longer than a second, a protection check is very cpu-intensive and it is unnecessary. Checking the dongle around once per minute is more than sufficient.

Adding Protection (DinkeyAdd)

DinkeyAdd Overview

The next stage is to lock your software to the Dinkey dongle by running DinkeyAdd.

DinkeyAdd can perform the following actions:

- **Program Your Dongles** - using the parameters specified in DinkeyAdd.
- **Lock Your Software** – it modifies your program so that at runtime it will look for a dongle that matches the parameters you have specified.
- **Create a Temporary Software Key** - if a customer has damaged their dongle this can be used as a temporary fix before a replacement dongle arrives.
- **Create a Demo Software Key** – enables you to securely create a fully functional time-limited demo of your product.
- **Encrypt Data Files** - you can encrypt data files that are accessed by your Windows Shell-protected program.

Most of this chapter will concentrate on the first two actions because they form the core of DinkeyAdd.

When you protect your software with DinkeyAdd it modifies your protected program so that it will search for the first Dinkey dongle that matches the SDSN, Product Code and dongle model specified in DinkeyAdd. In addition, for Plus and Net dongles it will also search the list of licences inside the dongle for the licence you specified in DinkeyAdd for the protected software.

How you protect your software depends on which model of dongle you use.

Dinkey Lite Protection

Lite dongles do not need to be programmed because they are pre-programmed with a Product Code which you specify when you order. You should use this same Product Code in DinkeyAdd when you protect your software (so that your protected program will search for dongles with this Product Code). The use of licences does not apply to Lite dongles.

Dinkey Plus/Net Protection

In this case you choose the Product Code yourself. This is programmed into the dongles along with the protection parameters for all the licences you specify. Each protected program is assigned a licence which indicates which protection parameters you want to use for that program. Normally you would use one licence per program but you can share licences between programs or use many licences from within a single program. Plus and Net dongles must be programmed before you send them to your customers.

Note - you do not need to specify the SDSN in the DinkeyAdd settings because it has already been burnt into your unique SDK. The SDSN is checked automatically by our code when it checks for the presence of a dongle.

How to Use DinkeyAdd

There are a number of different ways you can use DinkeyAdd. It is comprised of the following programs:

- **DinkeyAdd GUI** - this is the frontend GUI program that can be used to specify the protection parameters for your software. It calls a backend DinkeyAdd Module which performs the specified action.

- **DinkeyAdd Modules** - the backend modules that do all the work. We provide an API for these modules so that you can call them from your own code. For example, you might want to write a program for your sales team that can program your dongles based on a number of different configurations. We provide sample code to do this.
- **DinkeyAddCmd** - this is a dedicated command-line (terminal) program. You could use it, for example, to automate protecting your software as part of your build process.

The DinkeyAdd GUI is available for Windows and macOS. However the DinkeyAdd Modules and DinkeyAddCmd are available for all supported Operating Systems.

The DinkeyAdd GUI allows you to specify all the protection parameters required to lock your program(s) to your dongles. Once you decide which parameters you want to use then you should save these values to a DinkeyAdd Parameter File (**DAPF**). The DinkeyAdd Modules and DinkeyAddCmd all require a DAPF file as input. So, the DAPF file is the key to using DinkeyAdd.

There are two different formats of DAPF file: binary (.dapf extension) and JSON (.dapfj extension). The JSON format is a text format that allows you modify the protection parameters manually should you need to. You can also modify this file programmatically if your language supports JSON. The binary format is used by certain API calls of the DinkeyAdd Modules to modify the DAPF in memory. In many cases you won't need to modify the DAPF and so it doesn't matter which format you use.

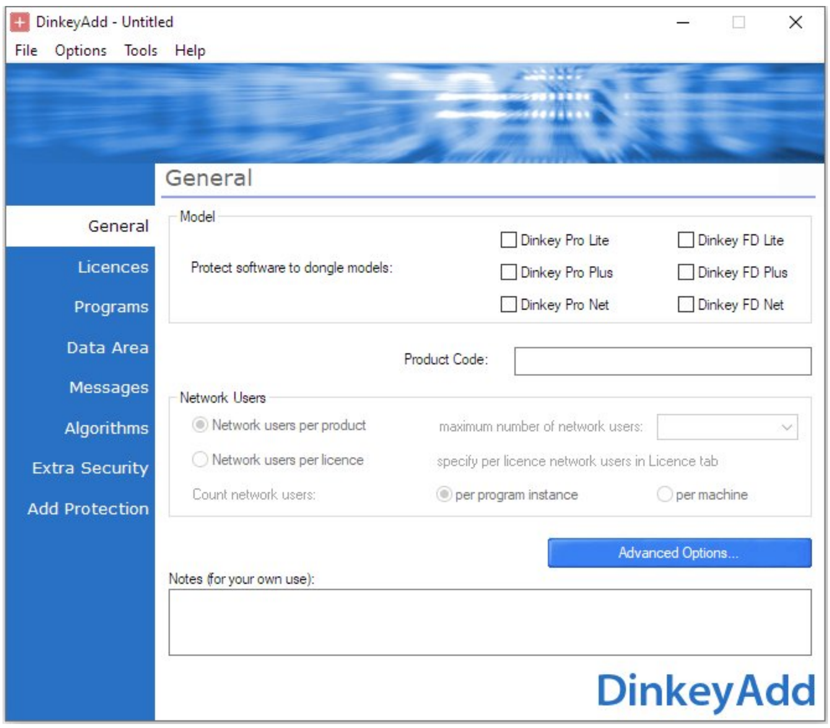
If you are running Windows or macOS then you can run the DinkeyAdd GUI program to create or edit a DAPF file. However, for Linux we do not provide this program and so you have to specify the protection parameters by manually editing the DAPF itself.

In the Quick Tour we supply a sample DAPF file in JSON format: hello.dapfj. You will see that the keys match the protection parameters in the DinkeyAdd GUI and are grouped into objects such as “Licences” and “Files” that correspond with the tabs in the DinkeyAdd GUI.

In the next section of this chapter we explain each protection parameter in detail. We will assume that you are using the DinkeyAdd GUI. However, if you need to manually modify the DAPF then you should be able to follow this section too in conjunction with this table listing the valid key values for the JSON DAPF file.

Protection Parameters

When you run the DinkeyAdd GUI it will come up with a screen looking like the one below.



General

Model

Protect software to dongle models: ☐ Dinkey Pro Lite ☐ Dinkey FD Lite ☐ Dinkey Pro Plus ☐ Dinkey FD Plus ☐ Dinkey Pro Net ☐ Dinkey FD Net

Product Code:

Network Users

[✓ Network users per product] maximum number of network users:

Network users per licence specify per licence network users in Licence tab

Count network users: [✓ per program instance] per machine

Notes (for your own use):

DinkeyAdd main screen

The various protection parameters that can be specified are described in detail in the section below.

General

Dongle Model

Please specify the dongle model(s) that you are locking your software to. You can specify more than one dongle model. In this case your protected software will work with any of the dongle models specified.

Product Code

This parameter can be up to 8 characters long (spaces are not allowed). If you are protecting your software to a Lite dongle then this value should match the Product Code pre-programmed into the dongle. For other dongle models this parameter can be any value you like. It is used by the dongle software to distinguish this product from another that you may protect in the future.

The Product Code is one of the values that identify your dongle (along with the SDSN and dongle number). Therefore, it cannot be changed later by DinkeyRemote.

Network Users

This parameter only concerns Net dongles. First you must decide whether you are going to limit the number of simultaneous network users per product or per licence. For example, if you are protecting 2 programs (each with a separate licence) and specifying a limit of 2 network users "per product", then at any one time the user can run either 2 copies of one program or 2 copies of the other or 1 copy of each program.

Protecting "per licence" means you can specify a separate limit for each licence. The total of all these limits must be less than or equal to the maximum value for that Net dongle. You can enter the "per licence" values by clicking on the Licence tab.

You also have the option to count the network users per instance of your protected program or per machine. In the latter case if a user runs multiple instances of your software on a single computer then it is only counted as a single network user.

Notes

This allows you to store your own notes for your own future reference. These notes are not programmed in the dongle or the protected program.

Advanced Options

For the vast majority of cases you will not need to modify these settings.

Dongle Number

This setting allows you to protect your software to a range of dongle numbers or to a specific dongle attached to the machine. The default setting is "Lock software to any dongle number" which is the recommended setting.

Note: The maximum dongle number is 4,294,967,294. You can also enter 'max' in the dongle range maximum.

Alternative Data Encryption Keys

For Plus and Net models three encryption keys are written to the dongle. These keys are generated based on the Product Code and SDSN of the dongle. In some cases you may want to decrypt files (for example) that were encrypted by another product. In this case please enter the Product Code for that product.

Lock Dongle to User's Computer

If you select this option (Plus and Net models only) then the first time you run any of the programs protected to this dongle, the dongle will be locked to the computer it is attached to. i.e. subsequently the protected software will not work on another computer even if the dongle is attached.

Do Not Allow USB Ports to be Shared across the Network

Some users install tools that allow USB ports to be shared across a network. If these tools are used then a non-Net dongle could be used on many computers across a network. (However, at any one time only one computer can claim the shared USB port). We have found that some of our customers want to allow this functionality and some do not. If you want to prevent these tools from being able to share a dongle in this way then please select this option. If this option is selected and we detect USB port sharing then the protection check will fail giving error 952.

Note – selecting this option has no effect under Linux.

Do Not Check for Software Keys at Runtime

If you select this option then your protected software will not search for a suitable Temporary Software Key or Demo Software Key if the correct dongle is not found.

Licences

[Plus and Net models only]

Use this tab to specify the licences that will be programmed into the dongle. As Lite dongles cannot be programmed this section does not apply to them.

A licence contains various protection parameters (such as expiry date, execution counter, etc...). Each licence is identified by a licence name. You can specify up to 255 licences. You assign each protected program with a licence in the “Programs” tab.

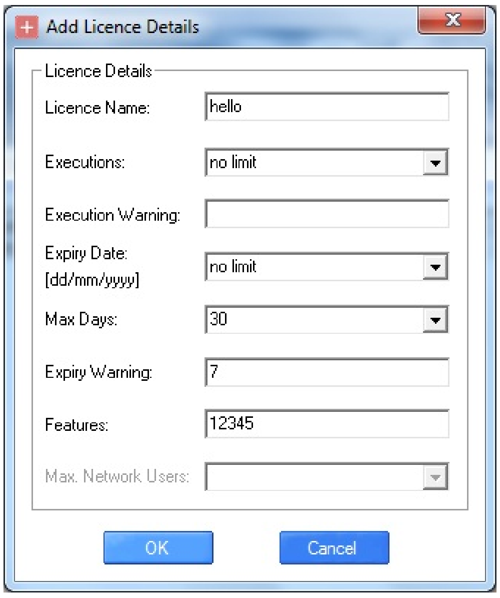
Normally you would specify one licence for each program that you want to protect. In this case a good name for the licence name is the name of the program you want to protect.

However, in rare cases you may want to specify different licences for some features of your program. For example, you may want to have a different execution counter for these features. In that case you could have your main program licence and other licences named “feature1”, “feature2” etc... At runtime you can use these different licences (instead of the one registered to the program) using the USE_ALT_LICENCE_NAME flag (API Method only).

Alternatively, you may want a few different programs to share the same licence (for example, if you are protecting a Java application then you need to use different modules for each OS but as they serve the same purpose you would want all these modules to share one licence).

However, you should be careful about using too many licences unnecessarily as you can only access them one at a time at runtime and there is no way to enumerate them using our API. Normally you will only need one licence per protected program. For example, if you want to enable/disable various features in your code then can use one licence and set the features value to represent which features are enabled. You could also consider using the data area to store licence information if the features value is too small.

The licence name can be anything you like but it should be something that makes your licensing model clear to understand.



Licence Name:	hello
Executions:	no limit
Execution Warning:	
Expiry Date: [dd/mm/yyyy]	no limit
Max Days:	30
Expiry Warning:	7
Features:	12345
Max. Network Users:	

The protection parameters that you can specify for each licence are:

Executions - the number of executions allowed for your program (or part of your program). This can be a number between 1 and 2,147,483,647 or 'no limit'. When calling the DDProtCheck function, the execution counter will be decremented only if the flags are set to DEC_ONE_EXEC or DEC_MANY_EXECS.

Execution Warning (Shell method only) – when the number of executions dips below this value then our software will display the "Execs Warning Message". Leave blank or 0 to never display any warning message.

Expiry Date - You can specify an expiry date for your program. After midnight (GMT) at the end of the expiry date our software will return an error saying that it is expired. (We use GMT so that the expiry date is consistent across time-zones).

Max Days - the number of days that your software will run after it is first run (after being protected by DinkeyAdd). It can be a number between 1 and 32766 or 'no limit'. When your software is first run the expiry date is calculated (unless you specify the DONT_SET_MAXDAYS_EXPIRY). If you specify an Expiry Date and Max Days then when your program is first run we will calculate which date is sooner and use that.

Expiry Warning (Shell method only) – when the number of days between the current date and the expiry date dips below this value then our software will display the "Expiry Warning Message". It will only display this message once per day, so as not to annoy the user. Leave blank or 0 to never display any warning message.

Features - 4 bytes that are for your own use. It can be used, for example, to determine which features of your program are enabled. It can be any number from 0 to 4,294,967,295 (hexadecimal: FFFFFFFF).

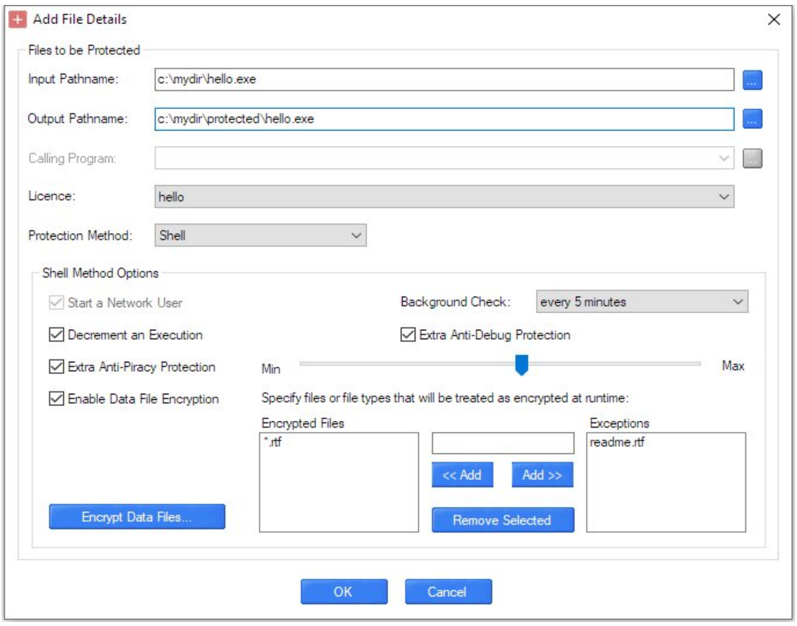
Network Users (Net models only) - this value limits the maximum number of simultaneous network users that can execute the specified program. (i.e. it is the "**per licence**" value and not the per product value. Please note that the sum of the limits of all the licences must be less than or equal to the maximum for that Net dongle).

Programs

Use this tab to specify the files that DinkeyAdd will protect. If you are using a Plus or Net model you can also specify the licence that is used when you do a protection check from this program (other licences can be used by specifying the USE_ALT_LICENCE_NAME flag in the DRIS – API Method only).

You can specify separate input and output filenames so that the original (input) file is not modified and the protected (output) file is written somewhere else.

You can also use wildcards in the filename if you want to protect a group of files e.g. specifying the input file *c:\mydir\*.dll* will protect all the dll files in the c:\mydir folder. If you use wildcards then the output path must be a folder. To add files to the list click the 'Add' button and another Dialog Box will open.


Add File Details

Files to be Protected

Input Pathname: c:\mydir\hello.exe

Output Pathname: c:\mydir\protected\hello.exe

Calling Program:

Licence: hello

Protection Method: Shell

Shell Method Options

☑ Start a Network User

☑ Decrement an Execution

☑ Extra Anti-Piracy Protection

☑ Enable Data File Encryption

Background Check: every 5 minutes

☑ Extra Anti-Debug Protection

Min — Max

Specify files or file types that will be treated as encrypted at runtime:

Encrypted Files: *.rtf

Exceptions: readme.rtf

DinkeyAdd - Add File Protection Parameters dialog

For **Input Pathname** you should enter the filename of the program you want to protect or use the browse button. You can specify a full path or a path relative to the current directory. You can specify wildcards (e.g. c:\mydir\*.exe) if you want to protect a group of files. The input file(s) will not be modified.

As an example, if you wanted to protect dpwin32.dll or another runtime module then you could specify the installation path for this parameter. e.g. c:\program files\Dinkey Pro 7.2.0\modules\dpwin32.dll.

The **Output Pathname** is the filename that DinkeyAdd will use to output your protected file. You can specify a full path or a path relative to the current directory. You can set the Input Pathname to be the same as the Output Pathname if you want the file to be modified. Otherwise the original input file is not modified. If you have specified wildcards in your input pathname then the output pathname should be a folder where the protected files will be placed.

If DinkeyAdd needs to overwrite an existing file it will give a warning. You can turn off this warning in the Options menu. The default action is to place the protected file in a sub-folder called "protected". However, you can modify this to any other folder. For example, if protecting dpwin32.dll or other runtime module you can output the protected dpwin32.dll file to the folder containing your application.

Calling Program - this setting can only be used if you are protecting a dynamic library (.dll in Windows, .so for Linux and .dylib for Mac). If you specify the calling program here, then at runtime the DLL will check that the program calling it matches the program you have specified here. Although this increases the security of the protection check there are two caveats: you can only have one program calling the protected DLL; if you modify the calling program then you will have to protect the DLL again to match the calling program. There is no need to update the dongle.

Licence (Plus and Net models only) – this setting allows you to specify the licence that the protected program uses.

Protection Method - the method of protection: "API" or "Shell" or "API & Shell". Note - you can only use the "API & Shell" method if you have linked one of the static object modules to your C/C++ or Delphi code. If you are calling a runtime dynamic library and you want to use both API and Shell Methods then you would have two program entries: the first would use the API method to protect the dynamic library and the second would use the Shell Method to protect your program.

Shell Method Options

Start a Network User – starts a network user during the initial protection check. You if you have protected your software to a non-Net dongle then this option has no effect.

Decrement an Execution – decrements the execution count during the initial protection check.

NB if you are using the API and Shell Method then you may not want to select these options so you can have more control within your code calling our API.

Background Check - this will cause the protected program to check the dongle at repeated intervals for the duration of the program.

Extra Anti-Debug Protection - this will implement some extra anti-debug techniques to your Shell-protected program. In rare cases you may receive specific error codes when running your protected program and we might recommend that you turn this option off.

Extra Anti-Piracy Protection - this will implement stronger and more widespread runtime encryption to your Shell-protected program. You can also specify the level of protection. However, this option can affect the speed of your protected program. If it runs much slower than expected you can try reducing the level of this option or even turning this option off. You can find out more about this option in the Shell Method section.

Enable Data File Encryption – select this option if you want to protect (i.e. encrypt) external data files that your Shell-protected Windows executable program accesses. You can specify which files (or file types, e.g. *.txt) are encrypted. You can also specify exceptions to these general rules. You will also need to encrypt the data files using the Data File Encryption Tool.

For example, if your protected program displays Rich Text Files you might want to encrypt these files so that users cannot copy this data. You should specify "*.rtf" in the list of **Encrypted Files**. It may be that there is a particular rtf file, say readme.rtf, that you do not want to encrypt. In this case you should list the file in **Exceptions**. Click on **Encrypt Data Files...** to invoke the Data File Encryption tool to encrypt the rtf files you want to distribute with your software. This tool can also be invoked from the main DinkeyAdd menu (Tools | Encrypt data files).

Note – this feature will affect file reads and writes across the entire process of the protected program. i.e. it will apply to files accessed from any DLLs loaded by your process as well as your executable. Therefore, even if you use a third-party DLL from your executable to read a certain data file type, if you Shell-protect your executable with this option then it will also affect data files read by the third party DLL. For this reason we do not allow Data File Encryption to be enabled when Shell-protecting DLLs.

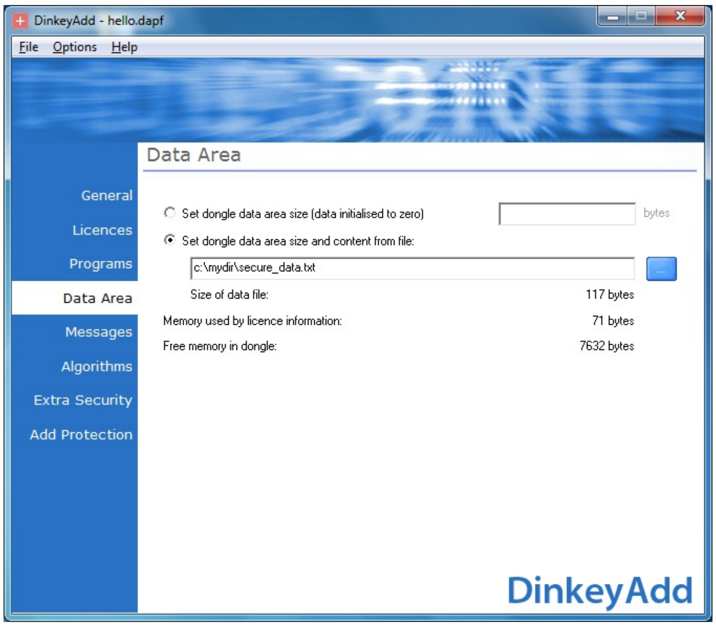
Data Area

[Plus and Net models only]

The memory usage of the Pro dongle is dynamically divided into two parts: that taken up by the parameters for the protected programs and that used by the data area. DinkeyAdd will automatically calculate how much memory is available in the dongle for the data area.

In general it is not recommended to set the size of the data area to its maximum value because this would prevent you from adding licences at a later date. Just use what you need; you can always change the size of the data area later by running DinkeyRemote if you wish.

You can initialise the data area by either specifying a data area size (the data will be initialised to zeros) or by specifying a file (the size of the data area will be set to the size of the file). At runtime when you call our API you cannot read/write beyond the size of the data area.



Set dongle data area size (data initialised to zero) bytes
[✓ Set dongle data area size and content from file:]
c:\mydir\secure_data.txt
Size of data file: 117 bytes
Memory used by licence information: 71 bytes
Free memory in dongle: 7632 bytes

DinkeyAdd – specifying dongle data area

Messages

[Shell Method only]

If you are using the Shell Method it will always do a protection check when your program first starts (or when it is first loaded for a DLL). It will always display a message depending on the result of the protection check (but if the message is blank then nothing will be displayed). These messages are stored with UFT-8 encoding.

The **Success Message** is displayed if there was no error checking the dongle.

The **Dongle Not Found Message** is displayed if no dongles are found attached to the computer. If a non-matching dongle was found then it will not display this message but the “Other Error Message”.

The **Expiry Date Reached Message** is displayed if the Expiry Date has passed.

The **Execs Expired Message** is displayed if the execution count has reached zero.

The **Expiry Date Warning Message** is displayed if the date passes the expiry warning date. If you use the symbol %1 in the message it will be replaced with the number of days left the user has.

The **Execs Warning Message** is displayed if the execution count is lower or the same as the Execution Warning value. If you use the symbol %1 in the message it will be replaced with the number of executions the user has left.

The **User Limit Exceeded Message** is displayed when the number of network users has exceeded the limit you specified.

The **DinkeyServer Not Found Message** is displayed when the protected program is protected to a Net dongle but cannot auto-detect DinkeyServer on the network.

The **Other Error Message** is displayed is there is an error but it falls into none of the categories above. Use the symbol %1 for the error number and %2 for the extended error number. You must display both of these values otherwise it may not be possible to diagnose a problem if you have one.

The **Background Check Error Message** is displayed if the background dongle protection check fails.

Note: you can include %L in any of the messages. This will be substituted by the licence name assigned to the protected program that displays this message.

Note: if your protected program is a Windows Service (or Linux daemon or a process that is running in a non-interactive account) then these messages cannot be displayed by the process. Instead we write them to a log file called *shellerrorlog.txt* which is located in the %AllUsersProfile%\Microcosm folder for Windows (<home>/.ddpro for Linux).

Algorithms

[Plus and Net models only]

This section allows you to specify up to 20 “user algorithms” and one “r/w algorithm” per dongle. This greatly improves the security that the dongle offers.

You can take formulas from your code and enter them as algorithms in the dongle. The algorithms only allow certain operations: addition, subtraction, multiplication, division, modulo (i.e. remainder after division) and the bitwise operators: AND, OR and XOR. You are allowed up to 8 independent variables, which must be whole integers (i.e. we do not use floating point arithmetic).

At runtime you can choose which “user algorithm” to execute using the alg_number field of the DRIS. You specify the values for variables a to h by setting the var_a to var_h fields in the DRIS. Our software will return an error at runtime if the algorithm has to divide by 0 or Mod 0 at any point during its execution. It is your responsibility to ensure that this does not happen. After the algorithm is executed the answer is returned in the alg_answer field of the DRIS.

You can then check that this value is correct thereby validating the dongle. Alternatively, and better, if the algorithm has been taken from a formula in your code then just use the answer that has been returned in your code. This makes it much more difficult for an attacker to patch your code because it doesn’t just boil down to a point where you are making a comparison and deciding whether to run or not. It is a much more complex process to follow.

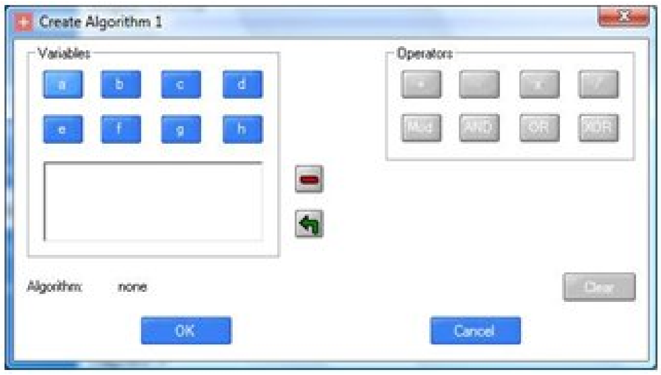
We assume the inputs and outputs are unsigned 32-bit integer values. When the algorithm is calculated in the dongle the resulting value will never ‘overflow’ – if the answer is too big for an unsigned 32-bit integer value then it will return the low 4-bytes of this answer. This is the default for languages such as C but not for Visual Basic. The sample code has been modified to take this into account.

The **R/W algorithm** is intended for a different purpose: to improve the security of data that is passed to or from our API (data is passed when you want to read or write the dongle data area or ask the dongle to encrypt / decrypt data that you provide). The R/W Algorithm has a restricted set of operations: addition, subtraction, XOR, AND, OR.

When you use a R/W algorithm you should use random values (in-between 1 and 16777215) as input values for variable a to h (this is why there are a restricted number of operations – we want to guarantee the answer to be always valid). The answer to the algorithm is used to generate encryption parameters in the code that encrypts your data. The sample code demonstrates how to do this.

To create an algorithm click on the “Add Algorithm” button.

The algorithm is displayed as you make up the algorithm. It must start with a variable and then alternate between operator and variable up to 11 times before ending with a variable. You can make up more complex algorithms by first making-up "composite variables" which can be used in the place of an ordinary variable. Composite variables are just algorithms. You add them to the "store" by clicking on the ↲ button.



DinkeyAdd – Create Algorithm Dialog

For example, to make-up the algorithm (a+b) * (c+d) * (e+f), first make up the algorithm a+b , then add it to the list of composite variables by clicking ↲ and then do the same for c+d and e+f. Finally make up the full algorithm by double- clicking a+b then x then c+d then x then e+f. Note: composite variables are not allowed in R/W algorithms.

After you have finished generating the algorithm you can add it to the dongle as a "user algorithm" or a "R/W algorithm" by clicking on the appropriate button.

If you select an algorithm and click on the "Generate Source Code…" button then sample code is generated in C/C++, Visual Basic, Delphi, C#, vb.net, Java.

Note – once you have programmed an algorithm into the dongle it cannot be displayed by DinkeyLook. This is another good reason for storing the DinkeyAdd parameters in a DAPF file.

Extra Security

[API Method only]

Encrypt DRIS

Tick this box to indicate that all calls to DDProtCheck will pass an encrypted DRIS. You must specify 3 encryption parameters - numbers between 1 and 255 - to vary the encryption. It does not matter which numbers you choose.

Click on "Generate Source Code…" to generate source code in different languages. You should patch this encryption code into your software to use to encrypt the DRIS. If you are using the sample code in the Samples folder then you should overwrite the existing CryptDRIS function with the new one that is generated. If you look at the sample code you will understand how this works. It improves security if there is some other code separating the calls to CryptDRIS from the call to DDProtCheck.

Encrypt data passed to API

Tick this box to indicate that all data passed to DDProtCheck and all data returned from DDProtCheck is encrypted. This data is passed either in the second parameter of DDProtCheck or pointed to by the rw_data_ptr field of the DRIS (depending on whether you use the USE_FUNCTION_ARGUMENT flag).

This time you can either specify 3 encryption parameters, or for even greater security, you can use the R/W algorithm. If you click "Generate Source Code" you will see how these values are used. If you use the R/W algorithm then at runtime you must specify the random variables used to get the R/W algorithm answer in the var_a…var_h fields of the DRIS.

Please look at the sample code to see how this works and is implemented for your lanaguage. It improves security if there is some other code separating the call to CryptApiData from the call to DDProtCheck.

Restrict dongle to only accept strongly encrypted DinkeyRemote codes

DinkeyRemote can generate two types of update code: strongly encrypted update codes and short codes. The strongly encrypted codes are very secure but long and not suitable for manually dictating to a customer over the phone. Instead they should be e-mailed. Alternatively the short codes are less secure (but still no-one has cracked them yet), but are short enough to be relayed to a customer on the phone.

It is more secure to restrict your dongle to only accepting the strongly encrypted codes but it does rule out the option of giving the code by phone to a user if their e-mail is not working.

Add Protection

Options

Lets you specify whether you would like to program the dongle (Plus and Net models only), add protection to your software or both of these options.

For example, if you are sending an upgrade of your software to a customer who already has a valid dongle for that product you then just need to "Lock software only". The Product Code and program names must match those in the users existing dongle otherwise your software upgrade will be searching for a different dongle!

Please note that DinkeyAdd can only program one dongle at a time and that your protected software will be encrypted differently each time.

Add Protection Now

Now that you have entered all the appropriate details you can protect your software (and/or program the dongle) by clicking on the *Add Protection Now* button.

At this point you may also want to keep a note of the protection details such as the dongle number, product code and to whom the dongles will be sent. You will need this information at a later date if you use DinkeyRemote or send out an upgrade.

We strongly recommend that you save the parameters used in DinkeyAdd to a DAPF file so that you can use these parameters in future (and also to help you remember what you have already done)!

Data File Encryption Tool

If you are using the Shell Method to protect your software and you have specified the Data File Encryption option then you will need to use this tool to encrypt your data files before you supply them to your users. If you encrypt the data files it makes them secure because the user can only use these files by running your protected software.

First you need to specify a list of data files to encrypt. You can specify the * wildcard to include file types as well as individual files. e.g. *c:\mydir\*.txt* will encrypt all text files in the c:\mydir directory. You should specify a separate file as the output (if you use wildcards in the input then the output should be a directory).

Click the Encrypt Files Now button to encrypt all the files in the list. This will overwrite any output files that may already exist.

The encryption algorithm is based on your SDSN and the Product Code in the DAPF file currently loaded. Therefore you cannot read encrypted files belonging to another Product (or another customer).

This tool is loaded in a dialog that is independent of DinkeyAdd so you can make changes to both at the same time. The list of encrypted files is stored in the DAPF file.

Note: you can automate the encryption of files by using the /e command line option in DinkeyAddCmd.

DinkeyAddCmd - Command Line

If you want to run DinkeyAdd from the command-line (terminal in Linux and Mac) then you should run the program DinkeyAddCmd, which is designed specifically for this purpose. For example you might want to do this to automate protecting your program as part of your build process. On Linux this is the only version of DinkeyAdd that we provide – there is no GUI program.

This is the usage syntax for the Windows version:

Add Protection / Programming dongles:

DinkeyAddCmd <dapf> [/a<n>] [/f<n>] [/d<filename>] [/q] [/s]

Encrypting Data Files:

DinkeyAddCmd <dapf> /e [/ei<input data file> /eo<output data file>] [/f<n>] [/q] [/s]

Converting DAPF files:

DinkeyAddCmd <dapf> /c<n> [/co<output file>] [/q] [/s]

Create a Temporary Software Key:

DinkeyAddCmd <dapf> /i<machineid> /x<dd-mm-yyyy> [/m<model>] [/g<dongle_number>] [/l<flags>] [/q] [/s]

Create a Demo Software Key Template:

DinkeyAddCmd <dapf> /y<maxdays> [/b<modelbitmap>] [/l<flags>] [/q] [/s]

Help:

DinkeyAddCmd [/?]

where:

dapf specifies the name of a DAPF file that you want DinkeyAddCmd to use. You can specify a full path or a relative path (the relative path is relative to the current working directory). The <dapf> parameter is required.

/a<n> means that DinkeyAddCmd will apply action <n> overwriting the action specified in the DAPF file. Possible values for <n> are:

1	program the dongle only
2	protect the software file(s) only
3	program the dongle and protect the software file(s).

/f<n> means that DinkeyAddCmd will only protect/encrypt the nth file in the list (0-based).

/d<filename> means that when DinkeyAddCmd successfully programs a dongle it will write the dongle number to the file specified. If the file already exists then it will be overwritten.

/e means that the Data File Encryption Tool will encrypt all the data files listed in the DAPF file (unless /f is specified). DinkeyAddCmd will not program a dongle or protect software if this switch is specified.

/ei<input data file> means that DinkeyAddCmd will encrypt the specified data file only and not the list specified in the DAPF file. Must be used with the /eo option.

/eo<output data file> specifies the output data file for the file encrypted using the /ei option.

/c<n> means that DinkeyAddCmd will convert the input DAPF file to the format specified:

1	binary format

2 JSON format

3 JSON format (minimal). See DAPF – JSON format for more details.

The default output filename is the input filename but with the extension of the new format (.dapf for binary, .dapfj for JSON). However, this default output filename can be overwritten by using the /co option.

/co<output file> specifies the output filename for the output DAPF file (must be used with the /c option).

/i<machineid> is the machine ID for the Temporary Software Key we want to create.

/x<dd-mm-yyyy> is the expiry date of the Temporary Software Key we want to create. The date must be exactly in this format.

/m<model> is the dongle model of the Temporary Software Key we want to create (see TEMP_SWKEY_INFO for values). If this option is not specified then we will use the model specified in the DAPF file (provided there is only one model selected – otherwise an error will occur).

/g<dongle_number> is the dongle number of the Temporary Software Key we want to create. If this option is not specified then a random dongle number is generated.

/l<flags> is the flags value as specified in the Software Key API functions. e.g. CreateTempSoftwareKey.

/y<maxdays> is the number of days from installation of the Demo Software Key until it expires.

/b<modelbitmap> is the bitmap of available dongle models for the Demo Software Key Template we are trying to create (see DEMO_SWKEY_INFO for values). If this option is not specified then we use the model specified in the DAPF file (provided there is only on model selected – otherwise an error will occur).

/q means that output messages will only be displayed if an error occurs.

/s means that no messages will be output (you can get the error code from the process exit code).

/? displays the full syntax of DinkeyAddCmd.

Note: DinkeyAddCmd will always overwrite output files.

Note: DinkeyAddCmd will always set the process exit code to either 0 (success) or an error code on failure.

For Linux and macOS DinkeyAddCmd has the same options but a slightly different syntax to reflect the way parameters are normally passed to command-line programs in those operating systems:

DinkeyAddCmd <dapf> [-a=<n>] [-f=<n>] [-d=<filename>] [-q] [-s]

DinkeyAddCmd <dapf> -e [-ei=<input data file> -eo=<output data file>] [-f=<n>] [-q] [-s]

DinkeyAddCmd <dapf> -i=<machineid> -x=<dd-mm-yyyy> [-m=<model>] [-g=<dongle_number>] [-l=<flags>] [-q] [-s]

DinkeyAddCmd <dapf> -y=<maxdays> [-b=<modelbitmap>] [-l=<flags>] [-q] [-s]

DinkeyAddCmd <dapf> -c=<n> [-co=<output file>] [-q] [-s]

DinkeyAddCmd [--help]

DinkeyAdd Module

If you want to write your own front-end GUI and incorporate all the functionality of DinkeyAdd then you can call the API of the DinkeyAdd modules. This might be useful, for example, if you want to write a program for your ordering team so that they can program dongles in certain configurations.

We supply 32-bit and 64-bit versions of this module for Windows, Mac and Linux. In Windows, for example, the 32-bit version is called **DinkeyAdd32.dll** and the 64-bit version is **DinkeyAdd64.dll**. There are some samples on how to call this module in the **Samples** folder of the SDK. We strongly recommend that you look at these samples and read the *readme.txt* file to understand how this API is used in your programming language.

Note – if you are using the DinkeyAdd Module to protect PDF files (Windows-only) then you need to have the file DinkeyAdd.pdf.dat in the same directory as the DinkeyAdd module.

A list of the API available in the DinkeyAdd module is specified below. The idea is that you get a handle to a DAPF file either through loading it from disk (LoadDAPFFromFile) or manipulating it in memory (ReadDAPFEx and LoadDAPF). You then use the handle to carry out the action required (e.g. AddProtection) and then unload the handle (UnloadDAPF).

The ReadDAPFEx and LoadDAPF functions are much more complicated than the LoadDAPFFromFile function as they use the complex binary DAPF structures in memory. An alternative to using these functions is to modify a JSON-formatted DAPF from your code and then use the simpler LoadDAPFFromFile function or execute DinkeyAddCmd.

DAPF Manipulation API Functions

dapf_handle **LoadDAPFFromFile** (IN string **dapf_file**, OUT int **error_code**)

This function loads a DAPF file from disk and returns a handle. If an error occurs the handle is set to NULL and *error_code* is set. Use this function if you want to use a DAPF file without any modifications (or without the simple modifications that the API allows to override the default behaviour).

int **ReadDAPFEx** (IN string **dapf_file**, OUT **DinkeyInfo** structure, OUT **LicenceInfo** structure, OUT **FileInfo** structure, OUT **Algorithm** structure, OUT **DataFileEnc** structure, OUT int **dapf_version,** OUT int **dapf_format**)

This function loads a DAPF file from disk and fills out the binary DAPF structures. This allows you to modify the DAPF file in memory. See binary DAPF Structures for more information about these structures.

The *dapf_version* parameter indicates the version of the DAPF file on disk (it is upgraded to the latest version in the DAPF structures returned to you). The *dapf_version* parameter can be useful if you want to save the file to an older version. It can be NULL if you do not use this parameter.

The *dapf_format* parameter indicates the format of the DAPF file on disk: 1 (DAPF_FORMAT_BINARY) or 2 (DAPF_FORMAT_JSON).

The return value of the function is the error code for this operation (0 = success).

dapf_handle **LoadDAPF** (IN **DinkeyInfo** structure, IN **LicenceInfo** structure, IN **FileInfo** structure, IN **Algorithm** structure, IN **DataFileEnc** structure, OUT int **error_code**)

This function takes the information from the DAPF structures (or arrays of structures), verifies it, and returns a handle on success (if an error occurs the handle is set to NULL and *error_code* is set). You can use this function with ReadDAPFEx or on its own if you want to make-up a DAPF file in memory from scratch.

int **WriteDAPFEx** (IN **dapf_handle**, IN string **dapffile**, IN int **dapf_format**, IN int **dapf_version**, IN int **flags**)

This function allows you to save the DAPF to disk.

The *dapf_format* specifies the format of the DAPF file you want to write: 1 (DAPF_FORMAT_BINARY), 2 (DAPF_FORMAT_JSON) or 3 (DAPF_FORMAT_JSON_MIN). The latter is a minimal format that only includes keys that differ from the default settings. This makes the file easier to read but potentially more difficult to modify.

The *dapf_version* specifies the version of the DAPF file you want to save to (0 means use the version specified in the DinkeyInfo structure).

If you set the *flags* value to 1 (FLAGS_LOSE_FEATURES) then this function will save the DAPF file to an older version even if it means losing some of the new features that have been set. The default action (i.e. flags = 0) is to give the error 2101 warning that these features will be lost.

The return value of the function is the error code for this operation (0 = success).

void **UnloadDAPF** (**dapf_handle_pointer**)

This function takes a pointer to the DAPF handle. It will release internal memory and NULL the DAPF handle.

Main API Functions

int **AddProtection** (IN **dapf_handle**, IN int **action**, IN int **file_num**, IN int **flags**, OUT uint **dongle_num**)

This function carries out the action specified by the DAPF file e.g. program a dongle or add protection to files (or both). This default action can be overwritten by the *action* parameter. Valid values are:

ACTION_USE_DAPF_SETTINGS	(0)	use the default action specified in the DAPF file
ACTION_PROGRAM_DONGLE	(1)	program the dongle only
ACTION_PROTECT_SOFTWARE	(2)	add protection to software files only
ACTION_PROGRAM_AND_PROTECT	(3)	program dongle and protect software.

File_num indicates that you want to protect the nth file in the (0-based) list specified in the DAPF file. The value -1 indicates that you want to protect all the files.

The flags value can be:

FLAGS_OVERWRITE_OUTPUT	(1)	overwrite any output files that might already exist.

Otherwise set to 0 (default behaviour).

If *dongle_num* is not NULL then it will be set to the dongle number if a dongle has been successfully programmed.

The return value is the error code for this operation (0 = success).

int **CreateTempSoftwareKey** (IN **dapf_handle**, IN **TEMP_SWKEY_INFO** structure, IN int **flags**)

Creates a Temporary Software Key using the inputs specified in the TEMP_SWKEY_INFO structure and the DAPF file.

The flags value can be:

FLAGS_OVERWRITE_SWKEY	(1)	overwrite an existing Software Key with similar parameters.
FLAGS_EXPIRY_DONT_CARE	(2)	create the Software Key even if the expiry date of a licence in the DAPF is earlier than the expiry date of the Software Key.

Otherwise set to 0 (default behaviour).

The return value is the error code for this operation (0 = success).

int **CreateDemoSoftwareKeyTemplate** (IN **dapf_handle**, IN **DEMO_SWKEY_INFO** structure, IN int **flags)**

Creates a Demo Software Key Template using the inputs specified in the DEMO_SWKEY_INFO structure and the DAPF file.

The flags value can be:

FLAGS_OVERWRITE_SWKEY	(1)	overwrite an existing Software Key with similar parameters.
FLAGS_EXPIRY_DONT_CARE	(2)	create the Software Key even if the max days of a licence in the DAPF is smaller than the max days of the Software Key.

Otherwise set to 0 (default behaviour).

The return value is the error code for this operation (0 = success).

int **EncryptShellDataFiles** (IN **dapf_handle,** IN int **file_num**, IN int **flags**)

Encrypts the shell data files listed in the DAPF specified.

File_num indicates that you want to protect the nth file in the (0-based) list specified in the DAPF file. The value -1 indicates that you want to protect all the files.

The *flags* value can be:

FLAGS_OVERWRITE_OUTPUT	(1)	overwrite any output files that might already exist.

Otherwise set to 0 (default behaviour).

The return value is the error code for this operation (0 = success).

Other API Functions

void **GetLastMessage** (IN string **message**, IN int **buffer_size**, OUT int **message_size**)

This function returns the message output by any of the API functions listed above (except UnloadDAPF). This is the message output normally displayed by DinkeyAdd (will occur on success and also on failure). Note - if you pass the message as NULL then the length of the message is returned in message_size.

int **GetLastExtendedError** (void)

This function returns the extended error, which will be set if an error occurs in any of the functions listed above. Note – the extended error is already included in the message returned by GetLastMessage so this function is not normally needed.

Runtime Behaviour of Your Protected Software

After you protect your software using DinkeyAdd, at runtime your software will do the following: search for a Dinkey dongle and return information on the first dongle found that matches the requirements specified in DinkeyAdd. i.e. matching dongle type, model, product code and SDSN. In addition, for Plus and Net dongles we also search the list of licences inside the dongle for the licence you specified in DinkeyAdd for the protected software (you can instead search for a different licence by using the USE_ALT_LICENCE_NAME flag).

You can also restrict the search to a particular dongle number if you set the MATCH_DONGLE_NUMBER flag when you do call DDProtCheck (API protection method).

Increasing Your Protection

Suggested Techniques

This chapter contains techniques to further improve the protection that the Dinkey system already offers. It is very important to read this chapter, as the security we offer is only as strong as your implementation of our software.

What follows applies to the API Method only. If you are using the Shell method then you can improve the security by also using the API method.

As discussed in the chapter "Understanding Software Protection" the weak part of the protection is the link between your software and the Dinkey software. The following techniques make it much more difficult for an attacker to patch your code or replace one of our protection modules.

Use the Shell Method as well as the API Method.

Both methods have their own security advantages. The Shell method encrypts your code so that it cannot be de-compiled. The API method enables you to implement many extra security measures outlined below.

Encrypt the DRIS and data passed to DDProtCheck.

If you encrypt the DRIS and data that is passed to or from DDProtCheck then it is extremely difficult for an attacker to see what values are being passed to our code. We recommend that the code used to encrypt/decrypt the DRIS and/or API data is separated from the call to DDProtCheck.

Put data from your software into the dongle data area.

Take out some data (e.g. a variable) from your program and place it in the dongle (Plus and Net models only). Every time you modify this data you will have to write it to the dongle. Every time you use this data you will have to read it from the dongle. (Remember that it will take longer to read/write to the dongle depending on how much data you read/write. It is best to use this technique on a variable that changes occasionally).

Using this technique you are effectively placing part of your program in the dongle. It is best to just use the data after it is read rather than specifically checking it has been changed to the correct value. This makes it very difficult for an attacker to patch your code since it does not reduce the security check to a single comparison. Also, it is difficult for the attacker to tell what the value should be just by looking at your code.

Take formulas from your code and add them to the dongle as algorithms.

Take out formulas from your code and add them to the dongle. This means you are effectively taking code out of your program and putting it in the dongle. As it is very difficult for an attacker to work out the algorithm then it is much more difficult for them to hack your code. Instead of checking the returned algorithm answer by executing the algorithm yourself it is better to just use the answer in your program. In this way the algorithm does not appear in your software at all.

If you do not have any complex algorithms in your code then you could make them more complicated when you add them to the dongle. E.g. suppose in your code your algorithm is just a + b + c + d. Then you can add the algorithm (a + b + c +d) XOR e XOR f XOR (g + h) to the dongle. After you get the answer you can then perform the operation XOR (g + h) XOR f XOR e in your code to get to the original answer.

Checksum our module (only if you are calling a DLL or Plugin).

If you are calling one of our DLLs or Plugins then we recommend that you calculate a checksum of this file to make sure that it has not been modified by a third party. Note – you will have to calculate the checksum and hard-code this into your software. If you upgrade our software or protect it differently then the checksum value will have changed. We do not give examples of code to do this because it may give attackers an idea of what checksum you will use. Try to use a checksum that doesn't just add up each byte or xor each byte.

Specify a Calling Program (only if you are calling a DLL).

If you are using one of our DLLs to protect your software then you can increase the binding between the calling program and your DLL by using this option. At runtime our DLL will check some features of the program calling it and see if they match your program when you ran DinkeyAdd. In this way you can stop attackers probing our DLL to find out what values it is returning. Of course, if you change your calling program then you will need to re-protect the DLL.

Call the dongle from many places throughout your program.

If you only call the dongle once then it is much easier for an attacker to isolate the call. Furthermore, they could remove the dongle after the protection check has been done and transfer it to another machine. However, please note that there is a limit to the number of write accesses to the dongle (see Technical Specifications). Each protection check using WRITE_DATA_AREA is a write access, as is decrementing the execution counter. If you use an expiry date then a protection check will also use a write access.

Check many of the values in the DRIS.

Instead of just checking the return code to DDProtCheck, we recommend that you check other values such as the product code, dongle number, algorithm answer and values in the data area.

Check the DRIS values many times, in separate places in your program.

If you can spread out these checks in your program and check them many times in different ways it will make attacking your code much more difficult

Remote Parameter Changing

Overview

[Plus and Net models only]

This section explains how you can change the protection parameters in a Dinkey dongle while the dongle is still at a customer site. To do this you need to use two programs: **DinkeyRemote** and **DinkeyChange**.

Firstly, you run DinkeyRemote and then enter the changes you want to make to the customer's dongle. DinkeyRemote will issue you with a special update code that you can give to your customer over the telephone or by e-mail. Your customer then runs DinkeyChange and enters the update code. The required changes will then be made to the dongle.

If you want to make sure that the customer has carried this out this procedure, DinkeyChange issues a unique **confirmation code** that you can check with the confirmation code generated by DinkeyRemote.

You can save the parameters specified in DinkeyRemote to a **DRPF** file. This can be opened the next time you run DinkeyRemote so that you don't have to type in the information next time. This can be particularly useful if you want to make many complicated changes (to the data area for example).

You must, of course, supply your customer with DinkeyChange if you intend them to use this feature.

DinkeyRemote Overview

This program generates the update code to send to your customer. It is structured in a similar way to DinkeyAdd:

- **DinkeyRemote GUI** - this is the frontend GUI program that can be used to specify the protection parameters you want to change in the remote dongle. It calls a backend DinkeyRemote Module which generates the update code.
- **DinkeyRemote Modules** - the backend modules that do all the work. We provide an API for these modules so that you can call them from your own code. For example, you might want to write a program for your sales team that can update an expiry date for a customer. We provide sample code to do this.
- **DinkeyRemoteCmd** - this is a dedicated command-line (terminal) program.

The DinkeyRemote GUI is available under Windows and macOS. However the DinkeyRemote Modules and DinkeyRemoteCmd are available for all supported Operating Systems.

The DinkeyRemote GUI allows you to specify all the parameters you want to change inside the dongle. Once you decide which parameters you want to use then you can save these values to a DinkeyRemote Parameter File (**DRPF**). The DinkeyRemote Modules and DinkeyRemoteCmd all require a DRPF file as input.

There are two different formats of DRPF file: binary (.drpf extension) and JSON (.drpfj extension). The JSON format is a text format that allows you modify the protection parameters manually should you need to. You can also modify this file programmatically if your language supports JSON. The binary format is used by certain API calls of the DinkeyRemote Modules to modify the DRPF in memory. In many cases you won't need to modify the DRPF file and so it doesn't matter which format you use.

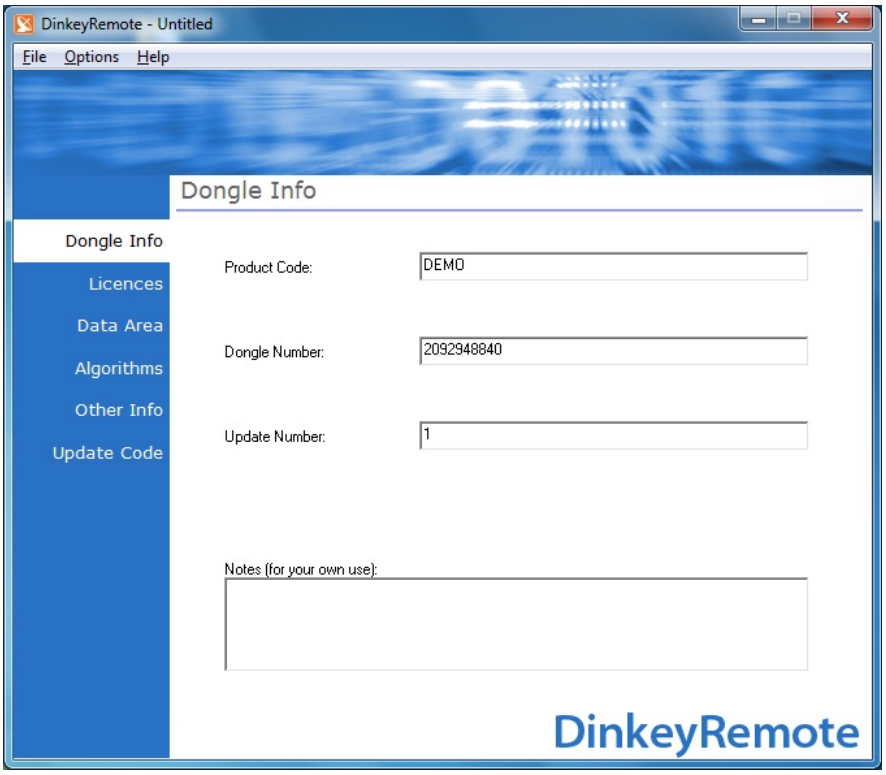
If you are running Windows or macOS then you can run the DinkeyRemote GUI program to create or edit a DRPF file. However, for Linux this program is not provided and so you have to specify the protection parameters by manually editing the DRPF itself.

In the Quick Tour we supply a sample DRPF file in JSON format: hello.drpfj. You will see that the keys match the protection parameters in the DinkeyRemote GUI and are grouped into objects such as "DongleInfo" and "Licences" that correspond with the tabs in the DinkeyRemote GUI.

In the next section of this chapter we explain each protection parameter in detail. We will assume that you are using the DinkeyRemote GUI. However, if you need to manually modify the DRPF then you should be able to follow this section too in conjunction with this table listing the valid key values for the JSON DRPF file.

DinkeyRemote

When you run the DinkeyRemote GUI you will see a screen similar to this:



Product Code: DEMO

Dongle Number: 2092948840

Update Number: 1

Notes (for your own use):

DinkeyRemote – main screen

Dongle Info

The first thing to do is enter the product code, the dongle number and the update number for the dongle concerned. You may have kept a record of the dongle number and product code when you first protected your programs for your customer. If not, then you can ask your customer to run DinkeyChange which will display a list of the dongles connected to their machine and the product code, the dongle number and the next update number for each one. They can copy and paste this information.

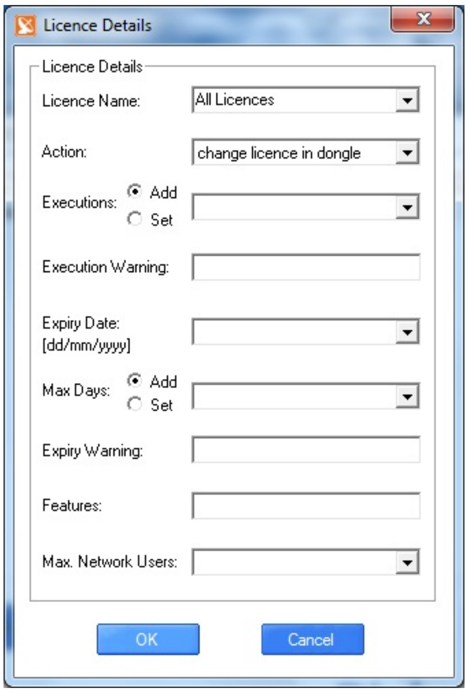
The **update number** starts at one and is incremented by one each time DinkeyChange has successfully run. This is to ensure that your customer can use your code once and once only. In fact, all DinkeyRemote update codes will be unique to that particular dongle and for the particular operation that you have specified. This ensures that nobody else can issue DinkeyRemote codes on your behalf.

Please note that since the dongle is characterised by its SDSN, product code, dongle number and update number these things cannot be changed by DinkeyRemote.

With DinkeyRemote you should specify what you want to change. Only enter information into those parameters that you want to change. If you do not want a certain parameter changed then leave that field blank.

Licences

In the Licences tab if you click on *Add to list* a new dialog box will appear.


Licence Details

Licence Details

Licence Name: All Licences

Action: change licence in dongle

Executions: [✓ Add] Set

Execution Warning:

Expiry Date: [dd/mm/yyyy]

Max Days: [✓ Add] Set

Expiry Warning:

Features:

Max. Network Users:

DinkeyRemote – Change licence parameters dialog

You have the option of changing parameters for all the licences in the dongle by specifying “All Licences”. If you have only one licence in the dongle you can also use “All Licences” as this makes the update code shorter. If you specify the Licence Name please note that it is case-sensitive.

As well as changing the parameters of a licence already in a dongle you can delete existing licences or add new licences to the dongle. If you delete a licence from the dongle then any programs that use it will fail the protection check. If you add a licence to the dongle then you will probably also want to protect a new program to go with this licence. Do this by running DinkeyAdd (specifying “Lock Software only”).

Please note that DinkeyChange will apply the changes in the same order that they have been added to DinkeyRemote.

Add Executions will add the specified number of executions to the execution count of that licence. You can select ‘no limit’ if you want this restriction to be removed.

Set Executions will cause the execution count of the licence to be set to the number specified. You can set this to ‘no limit’ also.

Execution Warning will change this value in the dongle. Only applies to the Shell method.

Expiry Date will set a new expiry date for that licence. You can select ‘no limit’ if you want any existing date restriction to be removed.

Add Max Days will add the specified number of days to the greater of the current date and the expiry date to produce a new expiry date. This can be set to no limit.

Set Max Days will allow the licence to be used for the specified number of days from the first time it has been run after DinkeyChange has been successfully executed. Any previously set expiry date will be overwritten. This can be set to ‘no limit’.

Expiry Warning will change this value in the dongle. Only applies to the Shell method.

Features lets you change features value.

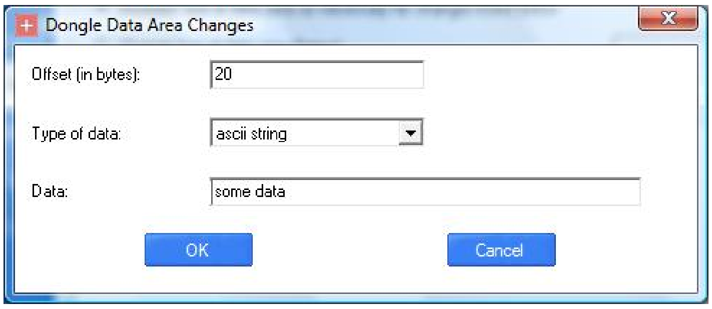
Max. Network Users will change the limit for the number of “per licence” simultaneous network users for this licence. If the dongle is currently using “per product” network users then this will fail.

Data Area

This tab allows you to change the data stored in the data area of the dongle.

You can also change the size of the data area (initially set to the size of the file used in DinkeyAdd to program the data area). Please note that if you reduce the size of the data area then you will lose data stored in the dongle. The default is to "increase the size of the data area as necessary" which will extend the size of the data area if any of the data changes you request necessitate this. In this case the data area is zero extended.

You can specify data area changes by clicking on *Add to List*:



Offset (in bytes): 20

Type of data: ascii string

Data: some data

DinkeyRemote – change dongle data area dialog

Offset is the offset in the data area where the new data will be written.

Type of data indicates the type of data you want to specify. It can be either from file, hexadecimal, ascii text or ascii string (this is the same as ascii text but has a null terminator at the end). You must then specify the file or data you would like to change. For large amounts of data you should use a file.

Algorithms

You can use this tab to add or overwrite existing user algorithms or the r/w algorithm in the customer's dongle. The functionality of the "create algorithm" dialog is the same as that for DinkeyAdd.

Note – it is possible using this technique to set user algorithm 4 (for example) when there is only 1 user algorithm in the dongle. In this case user algorithms 2 and 3 have not been set and so cannot be called at runtime. This is why the field in the DRIS is called max_alg_num and not number of algorithms. So, in this case max_alg_num would be 4 (because that is the largest valid algorithm number) but there are actually only 2 user algorithms in the dongle.

Other Info

Network Users Type – you can use this to change the type of network user protection to "per product" or to "per licence". You should only need to do this if you made a mistake in the initial programming of the dongle but have sent the dongle to the customer. Note the program the customer has must be protected using the same type of network user protection as that in the dongle.

Maximum 'per product' network users – specify the new value for per product network user maximum in the customer's dongle.

Count Network Users – you can change the way network users are counted to be "per program instance" or "per machine".

Change Last Date Used – you only need to set this value if the user has reported the error 419 (computer date has been tampered with). This could have been an accident or they could have been deliberately trying to beat the system. It is up to your discretion as to whether you issue this update code to reset the last date used. If you do decide to reset it you should set the last date used to today's date (or a date in the recent past).

Data encryption keys – set this if you made a mistake when programming the dongle and you need to change the product code the data encryption keys are based on.

Strongly encrypted update codes – you can change whether the dongle will in the future only accept strongly encrypted codes only or any type of update code.

Dongle locked to user's computer? – you can set whether the dongle will be locked to a user's computer (or not locked to any computers). If the dongle is set to be locked then, after the update code has been applied, it will lock to the first computer that the protected software protected is successfully run on.

Reset dongle so that it can be locked to another computer – if the dongle is already locked to a particular computer then you can reset the dongle so that it will lock itself to the first computer that the protected software is successfully run on.

Disable DinkeyOMS Dongle Blocking feature – this allows you to turn off the DinkeyOMS dongle blocking feature. You might want to do this if the user does not have an internet connection (and therefore the dongle blocking feature will not function correctly, eventually expiring with runtime error 974). Note that the dongle blocking feature can only be set by using DinkeyOMS.

Generate Update Code

DinkeyRemote can generate two types of update code: **strongly encrypted update codes** and **short update codes**. The strongly encrypted codes are very secure and it is recommended that you use this type of update code whenever possible. However, they are too long to dictate to a customer via telephone. If you cannot contact the customer via e-mail then you can generate a short update code.

It is possible to restrict a dongle to only accept strongly encrypted update codes in which case you do not have the option of generating short update codes. This increases the security of the update code but reduces the flexibility.

When you generate an update code, DinkeyRemote will output two files:

DinkeyRemote.log – a log file containing the summary of the changes you specified and the confirmation code. You can change the directory of the log file by choosing Options | Code Generation Settings from the menu. By default it is the Common Application Data Folder (C:\ProgramData for Windows Vista and higher).

UpdateCode.ducf - the update code file. By default this is written to the same output as the log file. However the output path and the update code file name can both be changed by specifying the **Output Path** and **Output file name**.

Other code generation settings include the option to generate an e-mail with the update code as an attachment and automatically copying the update code to the clipboard (for short update codes only).

DinkeyRemoteCmd - Command Line

If you want to run DinkeyRemote from the command-line (terminal in Linux and Mac) then you should run the program DinkeyRemoteCmd, which is designed specifically for this purpose. On Linux this is the only version of DinkeyRemote that we provide – there is no GUI program.

This is the usage syntax for the Windows version:

```
DinkeyRemoteCmd <drpf> [/d<dongle number>] [/u<update number>] [/o<ducf>] [/q] [/s]

DinkeyRemoteCmd <drpf> /c<n> [/co<output file>] [/q] [/s]

DinkeyRemoteCmd /?
```

where:

drpf specifies the name of a DRPF file that you want DinkeyRemoteCmd to use. You can specify a full path or a relative path (the relative path is relative to the current working directory). The <drpf> parameter is required.

/d<dongle number> - use the dongle number specified (rather than the dongle number in the DRPF file).

/u<update number> - use the update number specified (rather than the update number in the DRPF file).

/o<ducf> - use the DUCF file specified (rather than the DUCF specified in the DRPF file). The <ducf> parameter can include a relative path (relative to the output path set by DinkeyRemote) or can be an absolute path.

/c<n> means that DinkeyRemoteCmd will convert the input DRPF file to the format specified:

1	binary format
2	JSON format

3 JSON format (minimal). See DRPF – JSON format for more details.

The default output filename is the input filename but with the extension of the new format (.drpf for binary, .drpfj for JSON). However, this default output filename can be overwritten by using the /co option.

/co<output file> specifies the output filename for the output DRPF file (must be used with the /c option).

/q means that output messages will only be displayed if an error occurs.

/s means that no messages will be output (you can get the error code from the process exit code).

/? displays the full syntax of DinkeyRemoteCmd.

If you have to generate similar update codes for many different customers you can use the same DRPF file but specify the different dongle numbers and update numbers with the /d and /u command line options. You may also want to output to different DUCF files by using the /o command line option.

Note: DinkeyRemoteCmd will always overwrite output files.

Note: DinkeyRemoteCmd will always set the process exit code to either 0 (success) or an error code on failure.

For Linux and Mac DinkeyRemoteCmd has the same options but a slightly different syntax to reflect the way parameters are normally passed to command-line programs in those operating systems:

DinkeyRemoteCmd <drpf> [-d=<dongle number>] [-u=<update number>] [-o=<ducf>] [-q] [-s]

DinkeyRemoteCmd <drpf> -c=<n> [-co=<output file>] [-q] [-s]

DinkeyRemoteCmd --help

DinkeyRemote Module

If you want to write your own front-end GUI and incorporate the functionality of DinkeyRemote then you can call the API of the DinkeyRemote modules. This might be useful, for example, if you want to write a program for your ordering team so that they can update dongles in certain common situations without having to learn how to use DinkeyRemote.

We supply 32-bit and 64-bit versions of this module for Windows, Mac and Linux. In Windows, for example, the 32-bit version is called **DinkeyRemote32.dll** and the 64-bit version is **DinkeyRemote64.dll**. There are some samples on how to call this module in the **Samples** folder of the SDK. We strongly recommend that you look at these samples and read the *readme.txt* file to understand how this API is used in your programming language.

A list of the API available in the DinkeyRemote module is specified below. Similar to using the DinkeyAdd module, the idea is that you get a handle to a DRPF file either through loading it from disk (LoadDRPFFromFile) or manipulating it in memory (ReadDRPFEx and LoadDRPF). You then use the handle to carry out the action required (GenerateUpdateCode) and then unload the handle (UnloadDRPF).

The ReadDRPFEx and LoadDRPF functions are much more complicated than the LoadDRPFFromFile function as they use the complex binary DRPF structures in memory. An alternative to using these functions is to modify a JSON-formatted DRPF from your code and then use the simpler LoadDRPFFromFile function.

DRPF Manipulation API Functions

drpf_handle **LoadDRPFFromFile** (IN string **drpf_file**, OUT int **error_code**)

This function loads a DRPF file from disk and returns a handle. If an error occurs the handle is set to NULL and *error_code* is set. Use this function if you want to use a DRPF file without any modifications (or without the simple modifications that the API allows to override the default behaviour).

int **ReadDRPFEx** (IN string **drpf_file**, OUT **RemoteInfo** structure, OUT **LicenceInfo** structure, OUT **AlgChange** structure, OUT **DataChange** structure, OUT int **drpf_version**, OUT int drpf_format)

This function loads a DRPF file from disk and fills out the binary DRPF structures. This allows you to modify the DRPF file in memory. See binary DRPF Structures for more information about these structures.

The *drpf_version* parameter indicates the version of the DRPF file on disk (it is upgraded to the latest version in the DRPF structures returned to you). The *drpf_version* parameter can be useful if you want to save the file to an older version. It can be NULL if you do not use this parameter.

The *drpf_format* parameter indicates the format of the DRPF file on disk: 1 (DRPF_FORMAT_BINARY) or 2 (DRPF_FORMAT_JSON).

The return value of the function is the error code for this operation (0 = success).

drpf_handle **LoadDRPF** (IN **RemoteInfo** structure, IN **LicenceInfo** structure, IN **AlgChange** structure, IN **DataChange** structure, OUT int **error_code**)

This function takes the information from the DRPF structures (or array of structures), verifies it, and returns a handle on success (if an error occurs the handle is set to NULL and *error_code* is set). You can use this function with ReadDRPF or on its own if you want to make-up a DRPF file in memory from scratch.

int **WriteDRPFEx** (IN **drpf_handle**, IN string **drpffile**, IN int **drpf_format**, IN int **drpf_version**, IN int **flags**)

This function allows you to save the DRPF to disk.

The *drpf_format* specifies the format of the DRPF file you want to write: 1 (DRPF_FORMAT_BINARY), 2 (DRPF_FORMAT_JSON) or 3 (DRPF_FORMAT_JSON_MIN). The latter is a minimal format that only includes keys that differ from the default settings. This makes the file easier to read but potentially more difficult to modify.

The *drpf_version* specifies the version of the DRPF file you want to save to (0 means use the version specified in the RemoteInfo structure).

If you set the *flags* value to 1 (FLAGS_LOSE_FEATURES) then this function will save the DRPF file to an older version even if it means losing some of the new features that have been set. The default action (i.e. flags = 0) is to give the error 2101 warning that these features will be lost.

The return value of the function is the error code for this operation (0 = success).

void **UnloadDRPF** (**drpf_handle_pointer**)

This function takes a pointer to the DRPF handle. It will release internal memory and NULL the DRPF handle.

Main API Functions

int **GenerateUpdateCode** (IN **drpf_handle**, IN uint **dongle_number** , IN uint **update_number**, IN string **ducf_file**, IN string **logfile**, OUT UPDATE_CODE_OUTPUTS structure)

This function generates an update code with inputs taken from the DRPF file. These DRPF inputs can be overwritten by specifying non-zero values for the dongle_number, update_number parameters.

If *ducf_file* and *logfile* are non-NULL they will overwrite the default file names for the output DUCF file and logfile. You can specify full paths also if you want to change the location of these files.

See UPDATE_CODE_OUTPUTS for more information about the output structure. It contains the update code, update code file and the confirmation code.

The return value is the error code for this operation (0 = success).

Other API Functions

void **GetLastMessage** (IN string **message**, IN int **buffer_size**, OUT int **message_size**)

This function returns the message output by any of the API functions listed above (except UnloadDRPF). This is the message output normally displayed by DinkeyRemote (will occur on success and also on failure). Note - if you pass the message as NULL then the length of the message is returned in message_size.

int **GetLastExtendedError** (void)

This function returns the extended error, which will be set if an error occurs in any of the functions listed above. Note – the extended error is already included in the message returned by GetLastMessage so this function is not normally needed.

DinkeyChange

The DinkeyChange program will first scan for Plus and Net dongles attached to the computer and display the Product Code, Dongle Number and Update Number for these dongles.

The customer can now enter the update code. They can do this in either by manually typing the update code or by dragging and dropping the update code file. They must then click on "Make changes to dongle" to apply these changes.

The best method would be to associate DUCF file extension to DinkeyChange when you install your software. Then if the user double-clicks on the UpdateCode.ducf file it will launch DinkeyChange and apply the update code.

Apart from updating the dongle DinkeyChange can also perform the following actions:

1) If you want to see *all* the parameters stored in a customer's dongle then you can ask them to run Tools | **Diagnostics** which will generate a dlpf file with this information encrypted. If they send this information to you, you can view these parameters by opening the file using DinkeyLook.
2) If your customer is using Dinkey FD Lite and has accidently deleted the hidden .DO NOT DELETE.dat file on the flash disk and is getting error 442 (because they are running with restricted access rights) then they need to restore the dongle by running Tools | **Restore Dinkey FD Lite**. They need to be running DinkeyChange as administrator in order to do this.
3) Your customer can choose Tools | **Temporary Software Key...** if they need to download and install a temporary software key so that their protected software will continue to run for a limited period until a replacement dongle arrives for their damaged dongle. See the Software Keys chapter for more information. Under Linux you need to have root access for the first time you install a temporary software key.
4) Your customer can choose Tools | **Demo Software Key...** to download and install a demo software key if you want to distribute your protected software as a time-limited fully-functional demo. This step can be automated. See the Software Keys chapter for more information. Under Linux you need to have root access for the first time you install a demo software key.

The Linux DinkeyChange program should be run from the terminal. Please run DinkeyChange --help or see the readme.txt file in the Linux SDK for more information on command line parameters.

DinkeyChange Module

If you want to provide your own front-end GUI and incorporate all the functionality of DinkeyChange then you can modify your code to call a DinkeyChange Module.

We supply 32-bit and 64-bit versions of this module for Windows, Mac and Linux and also for Java. In Windows, for example, the 32-bit version is called **DinkeyChange.dll** and the 64-bit version is **DinkeyChange64.dll**. There are some samples on how to call this module in the **Samples** folder of the SDK. We strongly recommend that you look at these samples and read the *readme.txt* file to understand how this API is used in your programming language.

A list of the API available in the DinkeyChange module is specified below

int **DCGetInfo** (

IN int **type_mask**,
IN int **model_mask**,
IN string **prodcode_mask**,
IN int **array_length**,
OUT int **num_found**,
OUT int[] **type_array**,
OUT int[] **model_array**,
OUT string **prodcode_array**,
OUT uint[] **dongle_number_array**,
OUT int[] **update_number_array**
)

This function searches all the local USB ports on the user's machine and displays information on each of the dongles found. This function will only return information for dongles with your SDSN. You can restrict (mask) the search to dongles of certain types, certain dongle models or a specified product code.

Type_mask	Masks the search to the specified dongle types. Valid values are TYPE_MASK_PRO, TYPE_MASK_FD and TYPE_MASK_ALL.
Model_mask	Masks the search to the specified dongle models. Valid values are MODEL_MASK_LITE, MODEL_MASK_PRO, MODEL_MASK_NET, MODEL_MASK_ALL and MODEL_MASK_DEFAULT. The latter value is the default (search for Plus and Net models only).
Prodcode_mask	Masks the search to the product code specified. Use the empty string or NULL to search all product codes. Typically you would set this value to be product code of your product (unless you were writing a general program for more than one product).
Array_length	The lengths of the arrays passed (e.g. type_array, etc..). You can specify 0 if you just want to find out the number of dongles found.
Num_found	Returns the number of dongles found that meet the search criteria.
Type_array	Returns an array of the dongle types for all the dongles found. Valid values are TYPE_PRO and TYPE_FD.
Model_array	Returns an array of the dongle models for all the dongles found. Valid values are MODEL_LITE, MODEL_PLUS, MODEL_NET5, MODEL_NETU.
Prodcode_array	Returns an array of product codes for all the dongles found. Each element of the array is 9 characters long.
Dongle_number_array	Returns an array of dongle numbers for all the dongles found. Note – the dongle number is an unsigned integer.
Update_number_array	Returns an array of update numbers for all the dongles found.

The return value is the error code for this operation (0 = success).

int **DCGetDiagnosticInfo** (string **filename**)

This function writes diagnostic information to the filename specified. It appends the dlpf extension if necessary. This file can be sent to you and opened by DinkeyLook (it will show you many of the parameters inside each dongle attached). This function may take some time if many dongles are attached and/or they have large data areas. The return value is the error code for this operation (0 = success).

int **DCDoUpdateCodeString** (IN string **UpdateCode**, OUT int **confirmation_code**, OUT int **extended_error**)

This function will take a short update code, passed as a string and update the relevant dongle attached to the machine. If an error occurs then the extended_error value is also set. Otherwise the confirmation_code value is set. This should be displayed as a string of the hexadecimal value. The return value is the error code for this operation (0 = success).

int **DCDoUpdateCodeFromFile** (IN string **filename**, OUT int **confirmation_code**, OUT int **extended_error**)

This function will read a strongly encrypted code from the file passed in the filename parameter. It will then attempt to update the relevant dongle attached to the machine. If an error occurs then the extended_error value is also set. Otherwise the confirmation_code value is set. This should be displayed as a string of the hexadecimal value. The return value is the error code for this operation (0 = success).

int **DCRestoreDinkeyFDLite** (void)

This function will restore all Dinkey FD Lite dongles attached to the machine. This is needed if the user is getting runtime error 442.

int **DCGetMachineID** (OUT uint **machineID**, OUT int **extended_error**)

This function will calculate the machine ID of the user's computer and return it into the machineID parameter. This is the first stage for downloading a Temporary Software Key. The return value is the error code for this operation (0 = success). If an error occurs then the extended_error value is also set.

int **DCDownloadTempSoftwareKey**(IN uint **machineID**, OUT int **extended_error**)

This function will attempt to download a Temporary Software Key that you have generated for the machineID supplied. This is the second stage for downloading a Temporary Software Key. The return value is the error code for this operation (0 = success). If an error occurs then the extended_error value is also set.

int **DCDownloadDemoSoftwareKey**(IN uint **machineID**, IN string **ProductCode**, IN int **model**, OUT int **extended_error**)

This function will attempt to download a Demo Software Key for the product code and model specified onto the computer with the machineID supplied. The model parameter can take one of the following values:

SWKEY_MODEL_DEFAULT	(-1)	use the default dongle model. This assumes that the Demo Software Key Template was created with only one dongle model.
SWKEY_MODEL_PRO_LITE	(0)	Dinkey Pro Lite model
SWKEY_MODEL_PRO_PLUS	(1)	Dinkey Pro Plus model
SWKEY_MODEL_PRO_NET	(2)	Dinkey Pro Net model
SWKEY_MODEL_FD_LITE	(3)	Dinkey FD Lite model
SWKEY_MODEL_FD_PLUS	(4)	Dinkey FD Plus model
SWKEY_MODEL_FD_NET	(5)	Dinkey FD Net model

The return value is the error code for this operation (0 = success). If an error occurs then the extended_error value is also set.

DinkeyChange Command Line

Under Windows and macOS you would normally use either the DinkeyChange program or the DinkeyChange module. However, if you want to automate the installation of a Demo Software Key as part of your software installation package then you may want to create a script to run DinkeyChange from the command line. Under Linux DinkeyChange only exists as a terminal program.

Under Windows you can run DinkeyChange.exe as a command-line program only to install software keys. The syntax usage is described in the section How to Install a Demo Software Key.

Under macOS it is not possible to run the GUI DinkeyChange as a terminal program, so we created a terminal program called DinkeyChangeCmd64. The syntax is the same as for the Linux DinkeyChange and is documented below:

DinkeyChange <ducf>

DinkeyChange -i

DinkeyChange -d=<file>

DinkeyChange -r

DinkeyChange -m

DinkeyChange -t

DinkeyChange -e -p=<prodcode> [-l=<model>]

DinkeyChange –help

DinkeyChange --version

where:

<ducf file>

DinkeyChange will attempt to apply the changes specified in the ducf file.

-d=<file>, --diagnostics=<file>

DinkeyChange will write diagnostic information to the file specified.

-i, --info-only

Displays only information about the dongles detected.

-r, --restore-fd-lite

Restores a corrupted Dinkey FD Lite dongle (you need to be root).

-m, --machine-id

Displays the machine ID for the computer.

-t, --download-temp-swkey

Attempts to download a Temporary Software Key to the computer.

-e, --download-demo-swkey

Attempts to download a Demo Software Key to the computer. Use with -p (required) and -l (optional).

-p=<prodcode>, --swkey-product-code=<prodcode>

Specifies the Product Code for the Demo Software Key (use with -e).

-l=<model>, --swkey-model=<model>

Specifies the model (optional) for the Demo Software Key (use with -e). Values for model are the same as for the DCDownloadDemoSoftwareKey function.

--help

Displays options

--version

Display DinkeyChange version information.

Licensing Strategy

Developing a Licensing Strategy

Now that you understand the principles of the Dinkey Pro system you need to work out your software protection licensing strategy for your product. It is worth spending some time thinking about the best implementation for your needs because once this is fixed then it can be difficult to change in the future.
Sometimes developers have already created their own licensing system and use our dongles to implement that system. However, it is normally easier and more secure to start from scratch and use the functionality that is available in the SDK.
Note that when you protect your software with DinkeyAdd you are automatically locking it to your SDSN. This check is built-in to our code because the SDSN is embedded into your Dinkey Pro SDK and burnt into your dongles. Another developer will have a different SDSN and will not be able to detect your dongles.
The most common strategies are listed below along with the applicable protection methods and dongle models.

Basic Protection **(API, Shell) (Lite, Plus, Net)**

The simplest form of protection is to lock your software to your SDSN (this is done automatically) and a Product Code. You should use different Product Codes for each product you want to protect. Normally, you would not need to protect your software separately to each individual dongle, as this creates unnecessary extra work.

Time-Limited / Pay-per-Use **(API, Shell) (Plus, Net)**

If you want to set an expiry date or limit the number of executions of your application then you need to use a Plus or Net dongle. You can use DinkeyRemote and DinkeyChange to generate an update code to extend the expiry date or add/set the number of executions.

Feature-based **(API) (Plus, Net)**

If you want to control which features of your software are available to a customer then you need to use the API Method of protection with a Plus or Net dongle. You can use the features value in the licence as a bit field to indicate which features are available to your customer. When you check the dongle using our API then you can inspect the features value and enable the corresponding functionality in your software. If the features value (4 bytes) is not large enough then you could use the dongle data area to store this information. The features value can be updated remotely using DinkeyRemote and DinkeyChange.

Software Updates **(API, Shell) (Plus, Net)**

You may want to control which customers are allowed to use a new version of your software. You can achieve this with the API Method by setting the maximum version the customer is allowed to use in the dongle data area (or you could use the features value if you are not using this for another purpose). Alternatively you can achieve this with the API or Shell Method by protecting each new version to a new licence. For those customers that are allowed the new version, you use DinkeyRemote & DinkeyChange to add/replace this new licence in the dongle.

How to Use Licences

There are different strategies for using licences in the dongle:

- One licence per protected program: each protected program in your product has independent protection parameters (for example, a different execution count for each program).
- Many programs sharing one licence: many protected programs in your product share the same protection parameters (for example, they have the same expiry date).

- One program using many licences (API Method only): different parts of your software need independent protection parameters (for example, different features in your program have different execution counts). Note that you cannot enumerate licences in a dongle but if you know a licence name then you can check for this licence using the USE_ALT_LICENCE_NAME flag (by default your protected program will use the licence assigned to it by the DAPF file).

Network Dongles

About Network Dongles

When using the Lite and Plus dongle models your protected software will only detect dongles on the local machine. However, when using a Net dongle your protected software will search the whole network for a valid dongle, so instead of needing one dongle per machine you can have one dongle per network. Furthermore you can restrict the number of simultaneous network users of your protected software.

To protect your software to a Net dongle and also to specify the maximum number of simultaneous network users you need to use DinkeyAdd. You can also protect your software so that it will first look on the local machine for a Lite or Plus dongle and then look for a Net dongle. You can limit the number of simultaneous network users "per product" or "per licence" and also the way that network users are counted: "per program instance" or "per machine". See DinkeyAdd for more details on how to do this.

In order for the Net dongles to work correctly you need to install DinkeyServer on the computer that the dongle is attached to. On the client machines your protected software will automatically search the network for DinkeyServer which will communicate with the dongle on the server.

If you are using the API Method then you need to set the flags value to include START_NET_USER when you want to start a network user (this is generally done when the program is first started). The network user will automatically be freed when the program terminates so there is no need to use STOP_NET_USER unless you specifically want to stop the network user before the program has terminated.

Installing and Configuring DinkeyServer

You need to provide the Network Administrator with the Dinkey Dongle Network Server. It is named DinkeyServer.exe but you can rename it if you wish. It is specific to your SDSN, so will only serve products that belong to you.

To install DinkeyServer simply copy it to a directory on the machine that the Net dongle is attached to. This can be any machine on the network – it does not have to be a network server.

To configure DinkeyServer first run it and then select the appropriate options from the configuration dialog:

1. Choose to run DinkeyServer as an Application or as a Service (daemon under Linux).

   A Service starts automatically when the machine is booted and does not require a user to be logged in for it to run*. Starting DinkeyServer as a Service is the recommended setting for general use.

   An application requires a user to be logged-on for it to run. This may not always be the case on server machines. You may want to start DinkeyServer as an application if you want to test it.

2. The Network Configuration section allows you to configure the network settings that DinkeyServer will use. In Automatic mode, DinkeyServer will automatically choose the settings based on the best defaults combined with any existing configuration settings. This is the recommended setting. Custom mode allows you to manually specify the IP address and port that DinkeyServer will listen on.

*Under Linux if you want your daemon to automatically start when you boot up then you will need to create an init script to do this.

Note – Under Windows if you choose to run DinkeyServer as an application and want to have it start automatically when the user logs in, then you must create a shortcut to DinkeyServer in that user's 'Startup' folder and specify the appropriate command-line parameters to make DinkeyServer start automatically.

Note – DinkeyServer will list the Product Codes for each Net dongle attached to the computer.

To start DinkeyServer click the "Start" button. It will then either run as an application and launch the viewer screen or install itself as a service and run in the background. You will need to have Admin Rights to install a service.

Using DinkeyServer

The DinkeyServer Viewer

When DinkeyServer is running as an Application, it will display the DinkeyServer Viewer. This utility displays the status of DinkeyServer, the Product Codes it is serving and the network users currently connected to it. A DinkeyServer icon will appear in the Windows System Tray.

You can also open the server's log file using this utility. The log file records important events during the running of the server and may need to be viewed to diagnose problems.

When DinkeyServer is running as a Service, it will not display the viewer because Services cannot display Graphical User Interfaces (GUIs). However, you can still use the DinkeyServer Viewer by running DinkeyServer.exe with the /viewer command-line parameters.

Command Line Parameters

DinkeyServer can accept several command-line parameters which control its behaviour. These are as follows:

/s	Configures DinkeyServer to install/run as a Service (daemon).
/a	Configures DinkeyServer to run as an Application.
/q	Quiet mode. Only displays errors.
/u	Uninstalls the DinkeyServer Service and terminates the DinkeyServer viewer.
/t	Stops the DinkeyServer service and terminates the DinkeyServer viewer.
/showconfig	Displays the configuration information (e.g listen address, port). Linux only.
/viewer	Just display the DinkeyServer Viewer.
/logfile	Just open the DinkeyServer logfile.
/listen=IPADDRESS	Sets the IP Address on which DinkeyServer accepts connections (replace "IPADDRESS" with your chosen IP Address). You would only need to set this if your machine has more than one IP address.
/port=PORT	Sets the port on which DinkeyServer listens for connections (replace "PORT" with your chosen port number).
/?	Displays a help window documenting these command-line parameters.

Note: in Linux and macOS instead of using / for parameters you use – for short parameters and -- for longer parameters. e.g. –s and --viewer. Using --help will display the full syntax under Linux.

General Notes

Like any Service, DinkeyServer can be controlled through Windows Services Management (Control Panel | Administrative Tools | Services).

If you remove or add a dongle to the machine running DinkeyServer then you need to restart DinkeyServer in order for it to be detected.

If you have multiple network cards installed on your machine then the protected program may fail to automatically detect DinkeyServer (this can also be caused by virtualisation software, such as VMWare, installing "virtual" network adapters onto your system). In this case try disabling ones that don't connect to the LAN on which DinkeyServer runs.

If you update a network dongle using DinkeyChange (e.g. increase the number of network users) while DinkeyServer is running then DinkeyChange will notify DinkeyServer which will instantly update itself. In this case there is no need to restart DinkeyServer. If you reduce the number of network users then DinkeyServer will terminate some clients if the number of current network users exceeds the new limit.

Firewalls

Failure to correctly configure your firewall(s) could result in DinkeyServer being inaccessible by client machines.

- Your firewall must allow both TCP and UDP traffic through to DinkeyServer.
- Your firewall must allow DinkeyServer to accept incoming connections on the IP Address and Port that you chose when setting up DinkeyServer.
- Clients (your protected software) auto-detect DinkeyServer by performing a multicast to 239.255.219.183 and whichever port you configured DinkeyServer to listen on. Your firewall must not block traffic destined for this address/port.
- Under Windows the listen address and port DinkeyServer is using can be found by choosing Server | Server details from the viewer menu.

Like most server software, DinkeyServer accepts incoming connections by "listening" on a given IP Address and port. If the machine which runs DinkeyServer also runs firewall software (either the Windows Firewall or a third party filewall) then the firewall software will need to be explicitly told to allow the incoming connections to get through to DinkeyServer. i.e. you must add DinkeyServer as an exception to the firewall on the server (for Windows Firewall use Control Panel | Windows Firewall to do this).

Likewise, if a firewall exists on a machine between the machine running DinkeyServer and the client machines then this will need to be configured too.

Note – sometimes anti-virus programs can also include a firewall as part of the package.

Note – under Windows, DinkeyServer will automatically attempt to configure Windows Firewall when it is first run.

Log File

DinkeyServer maintains its own log file in which it records various important events during its execution. While the server is running, all events are recorded to this log file rather than being displayed on-screen. The most common reason for needing to view the log file is to begin diagnosing a problem.

You can view the logfile by running DinkeyServer with the /logfile command-line parameter. Alternatively, you can view it from the DinkeyServer Viewer (Windows only).

For Windows, the logfile is actually stored on disk in the "All Users" profile under the DinkeyServer\<SDSN> directories (where <SDSN> is your Software Developer's Serial Number). For macOS and Linux the logfile is stored in the <home>/.DinkeyServer/<SDSN> directory.

Overriding DinkeyServer Auto-detection

In very rare cases the customer may be using a network that does not properly support UDP multicast and so your protected program will not be able to auto-detect DinkeyServer on the network. Or, you may want to specifically tell your client-side protected software where to look for DinkeyServer.

This is possible by creating a <prodcode>.ini file (where <prodcode> is the Product Code of your protected program).

For the Windows operating systems, this file can be located in one of the following locations (we search in this order):

1) The folder of the program (or runtime module) you protected with DinkeyAdd.
2) The %AllUsersProfile%\<sdsn> folder. For example, on Windows 10 with a demo dongle it would be c:\ProgramData\10101.

For Linux and macOS operating systems the file is called <PRODCODE>.conf (the product code must be upper case) and can be located in one of the following locations (we search in this order):

1) The same directory as the protected program (for dynamic libraries it should be in the same directory as the program which is calling the library. This may not be that convenient in some cases).
2) In the (hidden) .ddpro directory in the user's home directory.
3) In the /etc/ddpro directory. You will need to be root to create or modify this file.

The <prodcode>.ini file takes the following format:

[ddpro]

server=<ip address>:<port>   OR   server=<machine name>:<port>

autodetect=FALSE

where the *server* key indicates **either** the ip address of the machine where DinkeyServer is located **or** the machine name and <port> indicates the value of the port you chose in DinkeyServer. You can also optionally indicate whether you would like our code to try to auto-detect the server if DinkeyServer was not found given the ip address and port you specified. The default value for this is FALSE. For example:

```
[ddpro]
server=192.168.1.4:32768
autodetect=FALSE
```

Listing the Current Network Users from the Client

Your customers can use the DinkeyServer Viewer to view all the network users that are currently connected. However, this can only be done from the machine that is running DinkeyServer, which is usually a server that customers may have limited access to.

In some cases it is useful for customers to view a list of current network users from their client machines. For example, if a customer is trying to run your protected application but the number of current network users has already reached the maximum then they may want to find out who these users are.

If you are using the API Method of protection then you can make an API call (to the protected runtime module, e.g. dpwin32.dll) to get this information. Note that you must have previously made a call to **DDProtCheck** in order for this function to work correctly.

This function communicates with DinkeyServer and finds information about the current network users. It will only return information for dongles with your SDSN and the Product Code of your protected program.

```
int DDGetNetUserList (
        IN string licence_name,
        OUT int num_net_users,
        OUT structure array net_user_info,
        IN int num_structs,
        OUT int ext_err
)
```

The function returns 0 on success, otherwise an error code is returned. See "Common Error Numbers" for a list of possible errors.

licence_name	Restrict the call to find out network user information for this licence. If this is NULL then (for "per product" network users) the function will return network user information for all licences in the dongle; OR (for "per licence" network users) the function will return network user information for the licence assigned to this program.
num_net_users	Returns the current number of network users.

net_user_info	An array of NU_INFO structures containing information on the current network users. See “Structures” for more information on the NU_INFO structure.
num_structs	The number of structures in your array.
ext_err	If an error occurred this may give extra useful information about the error.

You can see an example of this function being used in the sample code.

Software Keys

Types of Software Key

A Software Key is a purely software-based protection system that can be used in place of a dongle for a limited period of time. Software Keys have a number of restrictions but can be useful in certain specific circumstances. There are two different types of Software Key that can be created by the Dinkey Pro SDK.

A **Temporary Software Key** is designed to be used as a temporary replacement if a dongle is damaged so a customer can continue using their protected software until a replacement dongle arrives.

A **Demo Software Key** allows you to securely produce fully functional time-limited demos of your software.

Restrictions

Both types of Software Key have the following restrictions:

1) You need internet access to upload the Software Key from DinkeyAdd.
2) Your customer needs internet access to download the Software Key and also to run the protected program (we regularly check online to verify the authenticity of the software key to ensure strong security).

Note – we perform checks to detect whether the user has modified their computer date and time to try to make the Software Key last beyond the expiry date.

Note – for a Software Key that is of the FD type, the "fd_drive" returned in the DRIS is a folder on your computer that you can read and write to.

In addition to these restrictions each type of Software Key has its own restrictions (see below).

Temporary Software Keys

After you have distributed the dongle and protected software to your customer it may sometimes happen that the customer damages the dongle and it no longer works properly. In this case you will need to replace the damaged dongle with a new one. However, the customer will not be able run your protected software until they receive the replacement dongle. If the customer is in a remote location this could take some time.

The Temporary Software Key has been designed to solve this problem.

When you program the replacement dongle with DinkeyAdd you can also tell DinkeyAdd to create a Temporary Software Key and upload it to the internet. The customer can then run DinkeyChange and download this Software Key onto their machine. They can now run your protected program as if a dongle were attached to the machine (until the Temporary Software Key expires).

When the replacement dongle arrives they can attach it to their computer and the software will continue working. This minimises the disruption caused by having a damaged or malfunctioning dongle.

How to Create and Install a Temporary Software Key

You need to perform the following steps:

1) Ask the customer to run DinkeyChange and from the menu choose Tools | Temporary Software Key and ask them to send you the Machine ID that this dialog displays.

2) Launch DinkeyAdd on your computer and open the DAPF file that you used to program the replacement dongle. Choose Tools | Generate Temporary Software Key and enter the Machine ID and other parameters as requested. The expiry date indicates how long the Temporary Software Key is valid for (14 days is the maximum allowed). The dongle model should be the dongle model that this particular customer is using. The dongle number will usually be "default" unless the customer's dongle needs to have a specific dongle number. Click "Create Software Key" to upload the software key to the internet.
3) Ask the customer to run DinkeyChange and from the menu choose Tools | Temporary Software Key (the same as step 1). They should then click "Download Software Key". Note, for Linux systems you will need to be root the first time that a Software Key is installed on a computer.

The Temporary Software Key is now installed on the user's computer and they can run your protected program as if a dongle were attached to the machine.

Note – instead of using the DinkeyAdd and DinkeyChange GUI you can use the DinkeyAdd module API function *CreateTempSoftwareKey* to create the Temporary Software Key and the DinkeyChange module API functions *DCGetMachineID* and *DCDownloadTempSoftwareKey* to install it. The DinkeyChange sample code gives you an example of how to do this.

Restrictions

In addition to the general restrictions for Software Keys listed above the following restrictions apply to Temporary Software Keys:

1) The Temporary Software Key is valid for a maximum of 14 days from creation.
2) The number of Temporary Software Keys you can issue to a particular computer, for a particular Product Code, is restricted to 3 per year.

Demo Software Keys

It is common for software vendors to produce demo versions of their software so that potential customers can test its functionality. There are two popular types of demo:

1) **Restricted-use time-unlimited demo**. In this case you protect your software with our dongle using the API Method. If a dongle is not found then you can run in restricted "demo" mode. However, if the customer has purchased a dongle from you then your software will run with no restrictions. In this way you can use the same code for a demo and your full system.
2) **Fully functional time-limited demo**. In this case you can use a Demo Software Key so that your protected software can function for a limited period of time. If the customer decides to purchase your software then you can send them a dongle and your protected software will continue working.

You can use DinkeyAdd to create the Demo Software Key Template. It is called a template because once it has been created it can be installed on an unlimited number of computers – but only once on each computer.

How to Create the Demo Software Key Template

Launch DinkeyAdd on your computer and open your project's DAPF file. Choose Tools | Generate Demo Software Key Template and enter the parameters as requested.

Max Days from Installation – how long the Demo Software Key is valid on the customer's computer after installation.

Dongle Models – you should select which dongle model you want the Demo Software Key to be. DinkeyAdd only allows you to specify dongle models that are used by your DAPF file. You can specify more than one dongle model in which case you need to specify the model when the Demo Software Key is installed using DinkeyChange (see below).

Click "Create Demo Template" to upload the Demo Software Key Template to the internet.

How to Install a Demo Software Key

Ask the customer to run DinkeyChange and from the menu choose Tools | Demo Software Key. The customer needs to enter the Product Code. If you created the Demo Software Key Template with multiple dongle models then they also need to specify the dongle model they require. They should then click "Download Software Key".

The Demo Software Key is now installed on the user's computer and they can run your protected program as if a dongle were attached to the machine.

However, it is possible to automate the installation process. You can do this by calling a DinkeyChange module from your installation script and calling the function *DCDownloadDemoSoftwareKey*. The DinkeyChange sample code gives you an example of how to do this.

Alternatively you can call DinkeyChange.exe and specify the Product Code (and optionally dongle model) on the command line. This is the DinkeyChange command line usage for use with Demo Software Keys:

```
DinkeyChange.exe  /e  /p<product code>  [/l<model>] [/q] [/s]
```

where:

/e means that DinkeyChange will attempt to download a Demo Software Key.

/p specifies the Product Code for the Demo Software Key.

/l (letter l) specifies the model for the Demo Software Key. If this option is not specified then DinkeyChange will assume that the default model is being requested. Values for model are the same as for the *DCDownloadDemoSoftwareKey* function in the DinkeyChange module.

/q means that output messages will only be displayed if an error occurs.

/s means that no messages will be output (you can get the error code from the process exit code).

The DinkeyChange command-line syntax is slightly modified for Linux and macOS:

```
DinkeyChange  -e  -p=<prodcode> [-l=<model>]
```

Demo Software Key Online Management

You can access the Demo Software Key Online Management system from DinkeyAdd | Tools | Demo software key management page or through the URL https://www.microcosm.co.uk/swkey/manage.

You can use the management system to do the following:

- View a log of demo software key downloads for each template that you have created.
- View the number of demo software key installations you have remaining.
- Disable a Demo Software Key Template.
- Allow re-installations for specific customers.
- Specify a Development Machine ID that allows you test Software Keys on a specified computer.

Restrictions

The following restrictions apply to the Demo Software Key:

1) The Demo Software Key is valid for a maximum of 60 days from installation.
2) You can only install one Demo Software Key to a particular computer for each Product Code.

Testing Software Keys

We allow you to specify one computer that can be used to test the Temporary and Demo Software Keys with some of the restrictions removed. You select this computer by using the Software Key Online Management Portal and entering this computer's Machine ID as the Development Machine ID in the Settings page.

Temporary Software Key – you can create an unlimited number of Temporary Software Keys for a specific Product Code on the development computer (the normal limit is 3).

Demo Software Key – you will not be charged (i.e. the installations count will not decrement) if you download a Demo Software Key onto the development computer. Note – like normal you will still have to Allow Re-install for your machine ID using the Software Key Online Management Portal each time you want to install another Demo Software Key on this computer.

Summary of Differences

- You can automate the installation of a Demo Software Key. Installing a Temporary Software Key is a 3-step manual process.
- For a Demo Software Key you upload a Template and then many Demo Software Keys can be downloaded on different computers. For a Temporary Software Key you create the key for each specific customer.
- For a Demo Software Key you can install one key per Product Code per computer. For the Temporary Software Key you can install at most 3 keys per Product Code per computer per year.
- The Demo Software Key has an online management system.
- There is a charge for the Demo Software Key but the Temporary Software Key is free.

Other Uses of Software Keys

If your customer downloads your software online there can be a delay before they receive the dongle. You could create a Software Key so that the customer can use your protected software straight away until the dongle arrives.

DinkeyLook

DinkeyLook

This chapter covers an extra utility: DinkeyLook. This program will search all the USB ports in the local machine for Dinkey Pro and FD dongles and display information regarding their contents. It will only display information for dongles that belong to your SDSN.

For Lite dongles it will display the Dongle Number, Product Code and in-built Lite algorithm. For Plus dongles it will display the dongle number, update number, the product code and information for all the licences in the dongle (expiry date, executions etc…). It will also display the user data area (if one exists). For network dongles it will additionally display the number of simultaneous network users. However, there are some values it does not display (because once they are set they cannot be read). These values are: the user algorithms and the extra security settings.

As DinkeyLook can reveal sensitive information in your dongles it is intended as a tool for software developers and not for end users. If you want to view parameters that are stored in a customer’s dongle then you can ask them to run DinkeyChange and then choose Tools | Generate Diagnostics. This will create a diagnostic file that they should send to you. You can then open the diagnostic file that the user sends to you using DinkeyLook. It will display information on all the dongles attached to the customer’s machine just as if he were running DinkeyLook on his machine. Of course, the diagnostics in the file are heavily encrypted so that the user will not be able to modify or view these values.

The Linux DinkeyLook program should be run from the terminal. Run DinkeyLook --help or view the Linux SDK readme.txt file for more information on command line parameters.

Distributing Your Software

What to Send to Your Customer

When you produce your product CD or installation package along with your protected program you may include the following:

- **DinkeyChange** – This program can be used as a diagnostic tool for all models of dongles. It is also used to accept update codes generated from DinkeyRemote for Plus and Net dongles. It is a stand-alone program and does not need any other modules for it to work. For more information please see DinkeyChange.
- **DinkeyServer** – if you are distributing Net dongles then you need to distribute this program to customers. It is a stand-alone program and does not need any other modules for it to work. For more information on DinkeyServer please see DinkeyServer.
- If you Shell-protected a .NET program then we will generate a file called <filename>.dp32.dll or <filename>.dp64.dll (where <filename> is the name of the file you Shell-protected) or sometimes we will generate both these files. This file (these files) should be distributed and reside in the same folder as your protected program.

Of course you also need to distribute the protected runtime dll/shared library (if you use it) and the DinkeyChange dll/shared library if you use that.

Please do NOT send DinkeyAdd, DinkeyRemote (or DinkeyAdd, DinkeyRemote modules) or DinkeyLook to your customers as this represents a serious security risk.

If you are distributing protected software to Linux operating systems then you also need to include the **inst** and **uninst** bash scripts as users will need to run these before the dongle will be detected correctly (see Linux SDK notes for more details).

Updating Your Software

When you update your software you will want to protect it before sending it to your customer.

To do this simply run DinkeyAdd and load the DAPF file you used to initially protect this software. On the "Add Protection" tab choose "Lock Software only" and then click "Add Protection Now". Your software will now be locked to the dongle that the customer has – there is no need for the customer to return the dongle.

DinkeyOMS

DinkeyOMS Overview

DinkeyOMS is an **O**nline **M**anagement **S**ystem for the Dinkey Pro and FD range of dongles. It is a separate product that can be used to augment the Dinkey Pro system.

You can use it to program your dongles, assign dongles to your customers and easily manage and apply all your dongle updates. You can view the current state of each customer's dongle and also view a history of changes that have been made to each dongle. You can even update multiple dongles at the same time.

DinkeyOMS simplifies the dongle update process. You make the necessary changes online [illegible]he user's dongle is automatically updated. In order for this to work the end user needs to run the OMSRuntime service on each computer that their dongles are connected to. The computer needs to have internet access. The service waits until an update code is ready for the dongle and then applies it. Therefore, DinkeyOMS is very useful for licensing models that involve generating a high frequency of update codes - for example, subscription or pay-per-use licensing.

DinkeyOMS can also solve the "lost dongle problem": if a customer reports that a dongle is lost, damaged or stolen then how can you tell if they are telling the truth or just trying to get an extra licence? DinkeyOMS lets you block a dongle so that it cannot be used again (after a few days leeway).

You can use DinkeyOMS to link with an internet-based payment system so that your customers can purchase product upgrades (for example extra executions, new features or to extend the expiry date of your application) and the dongle is updated automatically without you having to manually generate an update code.

Notes

DinkeyOMS is primarily designed for Plus and Net dongle models. Although Lite dongles can be added to the DinkeyOMS system they cannot be updated or blocked.

DinkeyOMS requires that users have an internet connection (or at least temporary internet access every now and then).

DinkeyOMS is included in the Dinkey Pro SDK and is located in the DinkeyOMS directory. For more information please read the DinkeyOMS manual.

Structures

Notes

In all the structures listed below the size of each field is in bytes. The "int" type is a 32-bit signed integer and "uint" is a 32-bit unsigned integer. The "char" type is a 1 byte ascii character. Therefore all strings are null-terminated ascii strings – except for certain strings that are explicity marked as utf-8. Some languages do not support unsigned integers or ascii characters – but all these issues are dealt with in the sample code.

DRIS Structure

The Dinkey Runtime Information Structure (DRIS) is a structure that is used as an interface between your program and the DDProtCheck subroutine. The DRIS is structured as follows:

Field	Size	Type	In/Out	Description
header	4	char	In	Must be set to 'DRIS'.
size	4	int	In	Size of this structure.
seed1	4	int	In	Random seed for use in DRIS encryption and also Data encryption.
seed2	4	int	In	Another random seed (see above for description).
function	4	int	In	This tells the API what you want it to do. Only use one function number at a time. For a list see "Functions".
flags	4	int	In	Allows you to specify options to each function. You can use more than one by OR-ing the values together. For a list see "Flags".
execs_dec	4	uint	In	Decrement "executions" by the value specified. Must be positive.
data_crypt_key_num	4	int	In	Specifies which encryption key to use when encrypting or decrypting user data.
rw_offset	4	int	In	Offset to read/write data
rw_length	4	int	In	Length of data to read/write from data area
rw_data_ptr	4/8	pointer*	In	Address of data to read/write
alt_licence_name	256	char	In	Tells our API to proceed but using the licence specified. Must be a null-terminated ascii string. The licence name is case-sensitive.
var_a	4	int	In	Variable to pass to algorithm.
var_b	4	int	In	Variable to pass to algorithm.
var_c	4	int	In	Variable to pass to algorithm.
var_d	4	int	In	Variable to pass to algorithm.
var_e	4	int	In	Variable to pass to algorithm.
var_f	4	int	In	Variable to pass to algorithm.

var_g	4	int	In	Variable to pass to algorithm.
var_h	4	int	In	Variable to pass to algorithm.
alg_number	4	int	In	Specifies which user algorithm to execute.
ret_code	4	int	Out	Error code.
ext_err	4	int	Out	Extended error code.
type	4	int	Out	The type of dongle (1 = Pro, 2 = FD)
model	4	int	Out	The model of dongle (1 = Lite, 2 = Plus, 4 = Net 5, 7 = Net unlimited).
sdsn	4	int	Out	Software Developer's Serial Number (SDSN).
prodcode	12	char	Out	Product Code (ascii null-terminated)
dongle_number	4	uint	Out**	The dongle serial number.
update_number	4	int	Out	The update number (starts at 1).
data_area_size	4	uint	Out	The size of the dongle data area you have allocated (in bytes).
max_alg_num	4	int	Out	The maximum algorithm number.
execs	4	int	Out	Executions left.
exp_day	4	int	Out	Expiry day (number).
exp_month	4	int	Out	Expiry month (number).
exp_year	4	int	Out	Expiry year (number).
features	4	uint	Out	Feature
net_users	4	int	Out	Maximum number of network users
alg_answer	4	int	Out	The result of your algorithm using the variables specified in var_a…var_h
fd_capacity	4	uint	Out	The capacity in bytes of the data area in the dongle detected. Only valid for FD dongles. Value is currently set to ~8K but may change in the future.
fd_drive	128	char	Out	For Windows this is the drive letter of the Dinkey FD detected e.g. f:\. For Linux/macOS this is the mount name of the flash disk.
swkey_type	4	int	Out	The type of Software Key detected (0 = no Software Key detected, 1 = Temporary Software Key, 2 = Demo Software Key).
swkey_exp_day	4	int	Out	Software Key expiry day (if a Software Key has been detected)
swkey_exp_month	4	int	Out	Software Key expiry month
swkey_exp_year	4	int	Out	Software Key expiry year

*4 bytes for 32-bit code, 8 bytes for 64-bit code

**also used as an input if the MATCH_DONGLE_NUMBER flag is set.

NB If a value is "no limit" then it will be stored as –1.

NU_INFO Structure

Field	Size	Type	In/Out	Description
licenceName	256	char	Out	Licence that the network user is using.
userName	50	char	Out	The user name of the network user.
computerName	256	char	Out	The computer name of the network user.
ipAddress	16	char	Out	IP Address of network user.

DAPF – Binary Format

This section defines the layout of the binary DAPF file. It is comprised of a series of smaller structures. These are the same structures that are used with the ReadDAPFEx and LoadDAPF functions (in the DinkeyAdd Module) that allow you to modify a DAPF in memory. You only need to understand these structures if you are using these API functions. We provide sample code in C, C# and VB.NET on how to do this.

An alternative to using the binary DAPF file is to use the DAPF file with JSON format. This is a human-readable text file.

Here is the current structure of the binary DAPF file, starting with the **DinkeyInfo** structure:

Field	Size	Type	Description
Header	4	char	Must be 'DAPF'
Version	4	int	Version of this structure (currently 9).

SelectedModels	4	int	Models selected in DinkeyAdd:
			BIT0 = Dinkey Pro Lite
			BIT1 = Dinkey Pro Plus
			BIT2 = Dinkey Net
			BIT3 = Dinkey FD Lite
			BIT4 = Dinkey FD Plus
			BIT5 = Dinkey FD Net

ProdCode	12	char	Product Code (ascii null-terminated)
PerLicence	4	int	0 = per product network users 1 = per licence network users
MaxNetUsers	4	int	Per Product Network Users
RangeSettings	4	int	0 = any number 1 = dongle in Computer 2 = range specified
RangeMin	4	uint	Minimum dongle number in range
RangeMax	4	uint	Maximum dongle number in range
NumLicences	4	int	Number of licences.
NumProgs	4	int	Number of programs to protect
NumAlgs	4	int	Number of user algorithms to program in the dongle.
DataSizeType	4	int	1 = set the zero data size, 0 = use file to initialise data area
ZeroDataSize	4	int	Size of the data area in bytes
DataFile	256	char	Path to file for data area (0 = no file).
DataAreaSize	4	int	-1 indicates "file not found".
SuccessMsg	256	char	Shell message – successful protection check (utf-8 encoding).

DongleNotFoundMsg	256	char	Shell message – dongle not found (utf-8 encoding).
ExpiryErrMsg	256	char	Shell message – expiry date reached (utf-8 encoding).
ExecsErrMsg	256	char	Shell message – execution limit reached (utf-8 encoding).
ExpiryWarnMsg	256	char	Shell message – expiry date warning (utf-8 encoding),
ExecsWarnMsg	256	char	Shell message – execution warning limit reached (utf-8 encoding).
NetUsersMsg	256	char	Shell message – network user limit reached (utf-8 encoding).
OtherErrorMsg	256	char	Shell Message – other error occurred (utf-8 encoding).
BackgroundMsg	256	char	Shell Message – background check failed error message (utf-8 encoding).
NetNoServer	256	char	Shell Message – failed to connect to DinkeyServer (utf-8 encoding).
RWAlgorithm	12	char	Coded form of R/W Algorithm
SecOptions	4	int	Security Options: BIT0 = DRIS encryption BIT1 = API Data encryption BIT2 = API Data encryption with r/w algorithm BIT3 = only accept strongly encrypted update codes
SecDRISEncKey	4	byte	DRIS encryption parameters. NB the 4th byte is ignored
SecDataEncKey	4	byte	API Data encryption parameters. NB the 4th byte is ignored
ProtectionType	4	int	How to add protection: 0 = lock software & program dongle 1 = program dongle only 2 = lock software only
DataEncKeyProdCode	12	char	Alternative Product Code used to generate internal encryption keys. Normally set to 0. Null-terminated.
Notes	256	char	User notes (utf-8 encoding).
LockDongle	4	int	1 = lock dongle to the user's computer. 0 = do not do this.
Reserved1	4	int	(Used by Dinkey OMS).
NetPerMachine	4	int	0 = count network users per program instance. 1 = count network users per machine.
AdvancedOptions	4	int	Advanced Options: BIT0 = do not allow USB port sharing for non-Net dongles BIT1 = do not check for Software Keys
NumDataFileEnc	4	int	Number of entries in our data file encryption list

This structure is followed by one **LicenceInfo** structure for every licence:

Field	**Size**	**Type**	**Description**
LicenceName	256	char	Name of Licence.
MaxDays	4	int	-1 = no limit
ExpiryDay	4	int	-1 = no limit
ExpiryMonth	4	int	-1 = no limit
ExpiryYear	4	int	-1 = no limit
Execs	4	int	-1 = no limit
ExecsWarn	4	int	0 = no warning

DaysWarn	4	int	0 = no warning
Features	4	uint	Features value.
LicenceNetUsers	4	int	number of per licence net users (-1 = no limit).

This is followed by a **FileInfo** structure for each program to protect:

Field	**Size**	**Type**	**Description**
InputName	256	char	Input pathname for the file to Protect. Null-terminated ascii string.
OutputName	256	char	Output pathname for the file after it has been protected. Null-terminated ascii string. If you are using wildcards (*) for the InputName then this should be a folder. Otherwise it should be a file.
CallingProgram	256	char	Calling program for protected DLL (0 = none). Null-terminated ascii string.
LicenceIndex	4	int	1-based index in the array of Licence information of the licence associated with this program. 0 means “no licence”.
Method	4	int	0 = API, 1 = Shell, 2 = API & Shell
ShellFlags	4	int	As per DRIS flags. Valid flags: DEC_ONE_EXEC, START_NET_USER
Background	4	int	0 = no background, 1 = every 1min, 2 = 5 min, 3 = 10min, 4 = 30min, 5 =1hr
OtherShellOptions	4	int	BIT0: Data File Encryption is enabled, BIT1: Extra Anti-Debug is disabled, BIT2: Extra Anti-Piracy is disabled (deprecated – now use ExtraAntiPiracy field instead).
EncFileList	256	char	List of files (separated by ‘:’) that the Shell-protected program will treat as encrypted. Can include wildcards.
ExceptList	256	char	List of files (separated by ‘:’) that are exceptions to the EncFileList
ExtraAntiPiracy	4	int	0 = off or an integer from 1 (min) up to 10 (max)

This structure is followed by an **Algorithm** structure for every user algorithm specified:

Field	**Size**	**Type**	**Description**
Algorithm	12	char	Coded form of user algorithm

This structure is followed by a **DataFileEnc** structure for every entry in the Data File Encryption list:

Field	**Size**	**Type**	**Description**
InputName	256	char	Full pathname of file to encrypt (can include wildcards)
OutputName	256	char	Full pathname of output file (or directory for inputs with wildcards)

DAPF – JSON Format

The DAPF settings can alternatively be specified in JSON format. This section lists the valid members in such a file and the permitted values for each of these keys (if you need to understand what each key represents you can refer to the protection parameters in the DinkeyAdd chapter).

The easiest way to understand the JSON DAPF file structure is to look at an example file in a text editor (for example, you could use the hello.dapfj file used in the QuickTour).

A value of *null* for a key indicates that the key is not applicable for your project. If a particular key is not specified in your DAPF file then the default value is assumed (note that certain keys such as ProductCode are required and have no default). This allows you to specify JSON DAPF files with minimal content which allows for greater clarity when viewing in a text editor.

Key names are case-sensitive but string values are case-insensitive (the use of CamelCase for key names may be unusual for JSON but it is consistent with the rest of our SDK). Note – in JSON if a string contains a backslash then it has to be escaped with another backslash. e.g. for a Windows file path you would specify "c:\\dir\\file.txt" for example. All strings are utf-8 unless otherwise specified.

The JSON DAPF file can contain the following members:

Top-Level Member	**Key name**	**Default**	**Permitted values**
Notes		null	string
Models		-	array of one or more from the following list: "Pro Lite", "Pro Plus", "Pro Net", "FD Lite", "FD Plus", "FD Net".
ProductCode		-	string (max 8 ascii characters)
NetUserType		"per product"	"per licence", "per product"
ProductMaxNetUsers		null	null, "no limit", integer
CountNetUsers		"per instance"	"per instance", "per machine"
Advanced	DongleNumbers	"all"	"all", "attached", "range"
	DongleNumberRangeMin	null	integer (the smallest dongle number in the permitted range).
	DongleNumberRangeMax	null	integer (the largest dongle number in the permitted range).
	DataEncKeyProductCode	null	string (max 8 ascii characters) or null
	LockDongleToComputer	false	true, false
	AllowUSBOverNetwork	true	true, false
	AllowSoftwareKeys	true	true, false
Licences			*an array of objects with the following keys:*
	Name	-	string
	Executions	"no limit"	"no limit", integer
	ExecutionWarning	null	null, integer
	ExpiryDate	"no limit"	"no limit", string of format "yyyy-mm-dd"
	MaxDays	"no limit"	"no limit", integer
	ExpiryWarning	null	null, integer
	Features	0	integer
	LicenceMaxNetUsers	null	null, "no limit", integer
Files			*an array of objects with the following keys:*
	InputName	-	string
	OutputName	-	string

	CallingProgram	null	string
	Licence	null	null (for Lite dongles), string (must correspond to a Name value in one of the Licence objects)
	Method	-	“API”, “Shell”, “API & Shell”
	ShellOptions		*an object with the following keys:*
	DecrementExecution	-	true, false
	StartNetworkUser	-	true, false
	BackgroundCheck	null	integer (must be one of: 1, 5, 10, 30, 60)
	ExtraAntiDebug	true	true, false
	ExtraAntiPiracy	6	Integer (0 = off, 1=minimum, 10=maximum)
	DataFileEncryption	-	true, false
	DataFileTypes	null	array of files or file types that will be encrypted. e.g. [“*.txt”, “*.rtf”]
	DataFileExceptions	null	array of files that are exceptions to the ShellDataFileTypes array. e.g. [“readme.txt”]
DataArea	ZeroDataSize	null	null, integer
	DataFile	null	null, string
ShellMessages	Success	null	string
	DongleNotFound	-	string
	ExpiryError	null	string
	ExecsLimit	null	string
	ExpiryWarning	null	string
	ExecsWarning	null	string
	NetUserLimit	null	string
	OtherError	-	string
	Background	null	string
	NetNoServer	null	string
Algorithms			array of strings (see algorithms note below)
ExtraSecurity	DRISEncryptionKeys	null	null, array of 3 integer values
	DataEncryptionKeys	null	null, array of 3 integer values
	RWAlgorithm	null	string (see algorithms note below)
	SecureUpdatesOnly	false	true, false
Actions	ProgramDongle	-	true, false
	LockSoftware	-	true, false
DataFileEncryptionTool	*Files*		*an array of objects with the following keys:*
	InputName	-	string
	OutputName	-	string

Note that Advanced, DataArea, ExtraSecurity, Actions and DataFileEncryptionTool are objects; Licences and Files are arrays of objects and that Algorithms is an array of strings.

Algorithms

Algorithms are listed as strings that are made up from the following variables: a, b, c, d, e, f, g, h; and the following operators: Mod, AND, OR, XOR. The variables and operators are not case-sensitive. You can use brackets to specify an order to the operations – otherwise we assume that the order is read from left to right. e.g. if you specify *a AND b XOR c + d * e* then we interpret this as (((*a AND b) XOR c) + d) * e.*

DRPF – Binary Format

This section defines the layout of the binary DRPF file. It is comprised of a series of smaller structures. These are the same structures that are used with the ReadDRPFEx and LoadDRPF functions (in the DinkeyRemote Module) that allow you to modify a DRPF in memory. You only need to understand these structures if you are using these API functions. We provide sample code in C, C# and VB.NET on how to do this.

An alternative to using the binary DRPF file is to use the DRPF file with JSON format. This is a human-readable text file.

Here is the current structure of the binary DRPF file, starting with the **RemoteInfo** structure:

Field	**Size**	**Type**	**Description**
Header	4	char	Must be 'DRPF'
Version	4	int	Version of this structure (currently 3).
ProdCode	12	char	Product Code (ascii, null-terminated)
DongleNumber	4	uint	Dongle number
UpdateNumber	4	int	Update number
Notes	256	char	User notes (utf-8 encoding).
NumLicences	4	int	Number of licences to change.
NumAlgs	4	int	Number of user algorithms to program in the dongle.
NumDataChanges	4	int	Number of changes to the data area
DataAreaSize	4	int	New size of the data area (in bytes). -1 indicates 'increase as necessary'
PerLicence	4	int	0 = no change 1 = change to 'per licence' net users 2 = change to 'per product' net users.
MaxNetUsers	4	int	Maximum Per Product network users -1 = no limit -2 = no entry (for display purposes)
LastUsedDay	4	int	Last used date: day (0 = no change)
LastUsedMonth	4	int	Last used date: month (0 = no change)
LastUsedYear	4	int	Last used date: year (0 = no change)
DataEncKey	12	char	Product Code for data encryption keys (null terminated).
ChangeCodeType	4	int	Update Code type dongle will accept: 0 = no change 1 = accept only strongly encrypted codes 2 = accept all types of update codes
CodeType	4	int	Type of update code to generate: 0 = strongly encrypted update code 1 = short update code
OutputFileName	256	char	Name of the file to output update code. 0 = default (UpdateCode.ducf).
LockDongle	4	int	0 = no change 1 = lock dongle to computer

			2 = unlock dongle from computer
ResetDongleLock	4	int	0 = no change 1 = reset dongle so that it can be locked to a new computer (in fact the next computer the protected software is run on).
DongleBlocking	4	int	0 = no change -1 = remove the DinkeyOMS dongle blocking feature
NetPerMachine	4	int	0 = no change 1 = change to count 'per instance' 2 = change to count 'per machine'
OutputPath	256	char	Path of the output ducf file. 0 = default (log file path)

This structure is followed by one **LicenceInfo** structure for each licence you want to change (if any):

Field	**Size**	**Type**	**Description**
LicenceName	256	char	Name of licence to change. Null-terminated ascii string.
Action	4	int	0 = change, 1 = delete, 2 = add
AddSetExecs	4	int	0 = no change to execs, 1 = add, 2 = set
Execs	4	int	-1 = no limit
ExecsWarn	4	int	0 = no warning
ExpiryDay	4	int	-1 = no limit, 0 = no change
ExpiryMonth	4	int	-1 = no limit, 0 = no change
ExpiryYear	4	int	-1 = no limit, 0 = no change
AddSetMaxDays	4	int	0 = no change to max days, 1 = add, 2 = set
MaxDays	4	int	-1 = no limit
DaysWarn	4	int	0 = remove warning, -1 = no change
FeaturesFlag	4	int	0 = don't change features, 1 = change features
Features	4	uint	Features value.
LicenceNetUsers	4	int	number of per licence net users. -1 = no limit, -2 = no change

This structure is followed by an **AlgChange** structure for every algorithm you want to change:

Field	**Size**	**Type**	**Description**
Alg_number	4	int	Algorithm number (0 = r/w algorithm)
Algorithm	12	char	Coded form of user algorithm

This structure is followed by a **DataChange** structure for every data change you want to make:

Field	**Size**	**Type**	**Description**
Type	4	int	1 = file, 2 = hex data, 3 = ascii text, 4 = ascii string
Offset	4	int	Offset in dongle data area to change
Length	4	int	Length of data to change
data	256	char	Data to write OR file name of data to write (depending on type value).

DRPF – JSON Format

The DRPF settings can alternatively be specified in JSON format. This section lists the valid members in such a file and the permitted values for each of these keys (if you need to understand what each key represents you can refer to the protection parameters in the DinkeyRemote chapter).

The easiest way to understand the JSON DRPF file structure is to look at an example file in a text editor (for example, you could use the hello.drpfj file used in the Quick Tour).

A value of *null* for a key indicates that this key is not applicable for your project or that this property will not be changed in the update code generated for the target dongle. If a particular key is not specified in your DRPF file then the value of *null* is assumed (note that certain keys are required - for example all the DongleInfo keys). This allows you to specify JSON DRPF files with minimal content which allows for greater clarity when viewing in a text editor.

The JSON DRPF file can contain the following members:

Top-Level Member	**Key name**	**Permitted values**
Notes		string
DongleInfo	ProductCode	string (max 8 ascii characters)
	DongleNumber	integer
	UpdateNumber	integer
Licences		*an array of objects with the following keys:*
	Name	String (ascii)
	Action	“change”, “delete”, “add”
	SetExecutions	null, “no limit”, integer
	AddExecutions	null, “no limit”, integer
	ExecutionWarning	null, “remove”, integer
	ExpiryDate	null, “no limit”, string in the format “yyyy-mm-dd”
	SetMaxDays	null, “no limit”, integer
	AddMaxDays	null, “no limit”, integer
	ExpiryWarning	null, “remove”, integer
	Features	null, integer
	LicenceMaxNetUsers	null, “no limit”, integer
DataArea	DataAreaSize	null, integer
	DataChanges	*an array of objects with the following keys:*
	Offset	Integer
	File	null, string
	HexData	null, string
	Ascii	null, string (ascii)
	AsciiString	null, string (ascii)
Algorithms	RWAlgorithm	string (see note for JSON DAPF about algorithms)
	Algorithm<*N*>	string (see note for JSON DAPF about algorithms)
OtherInfo	NetUserType	null, “per licence”, “per product”
	ProductMaxNetUsers	null, “no limit”, integer

	LastUsedDate	null, string in the format "yyyy-mm-dd"
	DataEncKeyProductCode	null, string (max 8 ascii characters)
	SecureUpdatesOnly	null, true, false
	LockDongleToComputer	null, true, false
	ResetDongleLock	null, true, false
	DongleBlocking	null, "remove"
	CountNetUsers	null, "per instance", "per machine"
UpdateCode	OutputFileName	null, string
	OutputPath	null,string
	Type	"secure", "short"

Note that DongleInfo, Algorithms, OtherInfo, DataArea and Action are objects; Licences is an array of objects. DongleInfo contains information about the target dongle. Licences, DataArea, Algorithms and OtherInfo contain information about what you want to change in the target dongle and UpdateCode contains information about the update code itself.

In the key *Algorithm<N>*, N represents a number between 1 and 20 (inclusive) e.g. "Algorithm1".

TEMP_SWKEY_INFO Structure

This structure is used by the CreateTempSoftwareKey API in the DinkeyAdd module.

Field	**Size**	**Type**	**Description**
size	4	int	Size of this structure
machineID	4	uint	Machine ID (stored as a hex integer) of the computer the Software Key will be installed on.
exp_day	4	int	Software Key expiry day
exp_month	4	int	Software Key expiry month
exp_year	4	int	Software Key expiry year
dongle_model	4	int	Dongle model of Software Key (0 = Dinkey Pro Lite, 1 = Dinkey Pro Plus, 2 = Dinkey Pro Net, 3 = Dinkey FD Lite, 4 = Dinkey FD Plus, 5 = Dinkey FD Net)
dongle_number	4	uint	Dongle number (0 = default: random dongle number)

DEMO_SWKEY_INFO Structure

This structure is used by the CreateDemoSoftwareKeyTemplate API in the DinkeyAdd module.

Field	**Size**	**Type**	**Description**
size	4	int	Size of this structure
max_days	4	uint	Maximum number of days an installed Demo Software Key can be used from installation until it expires
models	4	int	Bitmap of valid dongle models (0 means use the default model)
			BIT0: Dinkey Pro Lite; BIT1: Dinkey Pro Plus; BIT2: Dinkey Pro Net
			BIT3: Dinkey FD Lite; BIT4: Dinkey FD Plus; BIT5: Dinkey FD Net

UPDATE_CODE_OUTPUTS Structure

This structure is used by the GenerateUpdateCode API in the DinkeyRemote module.

Field	Size	Type	Description
ShortCode	384	char	The update code as a string (if the update code generate is not a short update code then this field will not be filled-in)
UpdateCodeFile	256	char	The full path and filename of the output DUCF file containing the update code.
ConfirmationCode	8	char	The confirmation code.

Protection Modules

Reference Guide

A number of modules are supplied with the Dinkey dongle package. You must use one of them when protecting your code using the API method. What you will need depends upon the language and operating system that you are using (for more detailed information read the readme.txt notes in the sample code folder). The following table is a summary:

dpwin32.dll	for 32-bit Windows programs.
dpwin32debug.dll	for 32-bit C/C++/Delphi/etc... programs that you need to debug. Not intended for release. See notes in samples\c or samples\delphi for more information.
DPJava.dll	32-bit Windows module for protecting Java programs.
DinkeyPro.fmx	Filemaker module.
ddpro.pyd	Python module. NB each version of Python will need a separate runtime module.
dpcom32.dll	COM/Active X module (intended for AutoCAD but can be used with other languages if required – not recommended as the security is weaker).
dpwin32_coff.obj	32-bit static object module for Microsoft C/C++ and MingW.
dpwin32_coff_debug.obj	Debug version of dpwin32_coff.obj. Not intended for release.
dpwin32_omf.obj	32-bit static object module for C++Builder and Delphi.
dpwin32_omf_debug.obj	Debug version of dpwin32_omf.obj. Not intended for release.

There are also some 64-bit protection modules:

dpwin64.dll	for 64-bit Windows programs
dpwin64debug.dll	for 64-bit C/C++/Delphi etc... programs that you need to debug. Not intended for release. See notes in samples\c for more information.
DPJava64.dll	64-bit Windows module for protecting Java programs.
dpwin64.obj	64-bit static object module for Microsoft C/C++
dpwin64debug.obj	Debug version of dpwin64.obj. Not intended for release.
dpwin64_delphi.obj	64-bit static object module for Delphi XE2 and higher.
dpwin64_delphi_debug.obj	Debug version of dpwin64_delphi.obj. Not intended for release.

There are a number of modules available for calling DinkeyChange:

DinkeyChange.dll	for 32-bit Windows programs.
DinkeyChangeDebug.dll	for 32-bit C/C++/Delphi etc... programs that you need to debug. Not intended for release.
DCJava.dll	for 32-bit Java programs.
DinkeyChange64.dll	for 64-bit Windows programs.
DinkeyChange64Debug.dll	for 64-bit C/C++/Delphi etc... programs that you need to debug. Not intended for release.
DCJava64.dll	for 64-bit Java programs.

There are also equivalent modules for Linux:

dplin64.o	Static 64-bit library for Linux. (Protect the program it links to using the API Method).
dplin64.so	Shared 64-bit library for Linux. (Protect this file using the API Method).
dplin64debug.o	Debug 64-bit static library for Linux. For 64-bit C/C++ programs that you need to debug. Not intended for release. See notes in samples/c for more information.
dplin64debug.so	Debug 64-bit shared library for Linux. For 64-bit programs that you need to debug. Not intended for release. See notes in samples/c for more information.
libDPJava64.so	64-bit Linux module for protecting Java programs.
DinkeyChange64.so	64-bit Linux shared library for DinkeyChange.
DinkeyChange64Debug.so	64-bit Debug Linux shared library for DinkeyChange.
libDCJava64.so	64-bit Linux shared library for DinkeyChange using Java.

All these modules have equivalent 32-bit modules for Linux.

There are equivalent modules for macOS:

dpmac64.o	64-bit static library for Mac. (Protect the program it links to using the API Method).
dpmac64.dylib	64-bit dynamic library for macOS. (Protect this file using the API Method).
dpmac64debug.o	Debug 64-bit static library for macOS. For 64-bit C/C++ programs that you need to debug. Not intended for release. See notes in samples/c for more information.
dpmac64debug.dylib	Debug 64-bit dynamic library for macOS. For 64-bit programs that you need to debug. Not intended for release. See notes in samples/c for more information.
libDPJava64.jnilib	64-bit macOS module for protecting Java programs.
DinkeyChange64.dylib	64-bit macOS dynamic library for DinkeyChange.
DinkeyChange64Debug.dylib	64-bit Debug macOS dynamic library for DinkeyChange.
libDCJava64.jnilib	64-bit macOS module for DinkeyChange using Java.

All these modules have equivalent 32-bit modules that are available on request.

Error Codes

Overview

All our programs and runtime modules give error codes if a problem occurs. If you or your customers encounter any error codes then you can use the tables below to try to fix the problem. Better still, you can look at our Knowledge Base which has an up-to-date list of error codes and fixes. You can find it at this URL:

https://www.microcosm.com/kb

If this still does not help you solve the problem then please contact Microcosm using the Contact Us link from the Knowledge Base. Please describe the problem and quote both the error code and extended error code.

If your software does not display the error code or extended error code then you can force our software to display these errors by setting the **DD_ERRCODE** environment variable. To do this set DD_ERRCODE=ON. Setting DD_ERRCODE=OFF will suppress any error messages displayed by us. Removing this environment variable will return your system back to its original state. You can set this environment variable by going to Control Panel | System | Environment.

Alternatively, you could ask the customer to run DinkeyChange and choose Tools | Generate Diagnostics and send the output file to you. You can open this file with DinkeyLook which will display the error code and extended error code.

Common Error Numbers

These error codes may occur at runtime or when running other programs that access the dongles. These are the most common errors. **If you find an error that is not listed here then please try to find it using the Knowledge Base on our website: www.microcosm.com/kb**.

400	Could not allocate enough memory to proceed with this operation. If this error occurs then you must be very low on system resources. A reboot will probably solve the problem.
401	Our software cannot detect any Dinkey dongles.
402	Too many dongles attached. For example, if you want to program a dongle using DinkeyAdd it will insist on only having one dongle attached.
403	The dongle detected is not of the expected type. For runtime code you select which type to protect your software to using the DinkeyAdd | General tab.
404	The dongle detected is not of the expected model. For runtime code you select which model to protect your software to using the DinkeyAdd | General tab.
405	Trying to use a demo dongle with non-demo software.
406	Trying to use a non-demo dongle with demo software or software protected by the demo DinkeyAdd.
407	The dongle detected belongs to another company (has a different SDSN).
408	The dongle serial number detected is not in the range specified in DinkeyAdd | General | Advanced (or you used the MATCH_DONGLE_NUMBER flag and specified a dongle number that differs from the dongle detected).
409	The dongle detected has not been programmed with DinkeyAdd.
410	The dongle detected does not have the expected Product Code. For runtime code you select the Product Code

	using the DinkeyAdd | General tab.
411	Cannot find the licence (associated with your protected program) in the dongle detected. You associate your protected program to a licence in the *Programs* tab of DinkeyAdd.
412	DRIS not passed correctly in the call to DDProtCheck. Please view our sample code for examples on how to do this correctly.
413	The program you are running has not been protected by DinkeyAdd. If you are using the API method then you need to protect dpwin32.dll and not your application.
414	The DRIS structure passed to DDProtCheck is too small. You have probably not correctly set the size field in the DRIS.
415	The DRIS structure passed to DDProtCheck is too large. You could be using a version of the DRIS file that is newer than the module you are calling. Alternatively you have not correctly set the size field of the DRIS.
416	The Operating System detected is not supported. Dinkey Pro is not compatible with Windows 95 or Windows NT4.
417	Bad parameter in DRIS structure. Can also be caused if you are encrypting the DRIS in your code but did not specify DRIS encryption in DinkeyAdd (or vice versa).
418	You have set at least 2 flag values in the DRIS that are contradictory.
419	The clock has been tampered with (either put forwards or put backwards). The user should first set the date and time to the correct value. If they still get this error then they need to receive a DinkeyRemote code to reset the "Last Date Used". Note, if you previously used "max days" to set an expiry date then you should check the expiry date is correct and/or set the max days value in the update code.
420	The execution counter has reached 0.
421	There are not enough executions left to decrement by the specified amount. This error was returned and no executions were decremented.
422	Expiry date reached.
423	Too many network users – the limit for the number of simultaneous network users has been reached.
424	Trying to read or write beyond the end of the dongle data area. Note that the dongle data area is set to the size of the data file you specified in DinkeyAdd. You can increase the data area size by generating an update code using DinkeyRemote.
425	Trying to write/read or encrypt / decrypt too much user data. For speed reasons we limit this to 1K. If you want to encrypt or decrypt more data then you can make more than one call.
426	The user algorithm that you want to execute does not exist. i.e. the alg_number field in the DRIS specifies an algorithm that does not exist in the dongle detected.
427	The algorithm that you specified with the variable values that you specified gives a 'Divide by Zero' error. This can also occur if you perform the operation X Mod 0.
428	This feature has not been implemented yet.
429	The alt_licence_name field specified in the DRIS is not null-terminated.
430	Calling program is not valid for the protected DLL.
431	A DLL that the Shell-protected program depends on does not exist on the user's machine.
432	The Dinkey FD Service (for original Dinkey FD dongles) needs to be installed as the user is running under Windows 2000 or XP with restricted access rights.
433	Requesting encryption from php (or asp.net etc...) that is not supported by the DinkeyWeb runtime module.
434	Need to upgrade DinkeyWeb module to support newer php (or asp.net etc...) sample code.
435	DinkeyServer not detected on the network. Either DinkeyServer is not running or it is being blocked by a firewall. If you have protected a Universal Windows Platform app then you could get this error if you have not enabled the "Private Networks" capability.
436	Can no longer connect with DinkeyServer. It may have terminated or the network may be down.
437	Your protected software uses a more recent version of the software than DinkeyServer. You need to upgrade

	DinkeyServer to a more recent version.
438	Error terminating a network user (e.g. one may not have been started)
439	No network dongles with the correct Product Code have been detected by DinkeyServer. If you have recently attached a dongle to the dongle server then you need to restart DinkeyServer for it to be recognised.
440	A network dongle with the correct Product Code has been detected but the program running is not in the list of protected programs in the dongle.
441	DinkeyServer is busy. If there are lots of requests to start a network user at the same time the server may be too busy to reply. If you receive this error then please display an appropriate message, wait and then try again.
442	The user has deleted the hidden .DO NOT DELETE.dat file from the flash disk of a Dinkey FD Lite dongle. The dongle will still work if the user is running as administrator. The dongle needs restoring by running DinkeyChange and choosing Tools | Restore Dinkey FD Lite or calling the DCRestoreDinkeyFDLite function in the DinkeyChange module.
443	The <prodcode.ini> (or <prodcode.conf> for Linux/macOS) server entry Ipaddress/machine:port has bad format or machine name does not resolve to a valid IP address.
444	The flash disk part of Dinkey FD has not been mounted by Linux/macOS. Change your settings so that this happens.
445	The function you are using in the DRIS is not supported by the dongle detected. E.g. trying to write data to a Lite dongle.
446	You have protected your software specifying API Data encryption with the r/w algorithm but there is not a r/w algorithm in the dongle.
447	You have tried to start more than one network user for the same program within the same process. This is not allowed.
448	Calling DDGetNetUserList without calling DDProtCheck successfully for a network dongle.
449	The licence specified in the call to DDGetNetUserList is not found in the dongle.
465	Under Linux this error means that you need to run the “inst” script. See the readme.txt in the Linux SDK for more details.
504	You can receive this error if you have protected a Universal Windows Platform app to a local dongle. You can only protect these apps with network dongles.
922	The Software Key detected has expired.
923	The Software Key detected is for the wrong type of dongle (e.g. FD instead of Pro)
924	The Software Key detected is for the wrong model of dongle.
925	Trying to use a Software Key (created with the demo SDK) with non-demo SDK.
926	Using the demo SDK but detecting a Software Key (created with the non-demo SDK).
927	The Software Key belongs to another company (has a different SDSN).
928	The Software Key detected has a dongle serial number that is not in the required range.
929	The Software Key detected has a different Product Code.
932	We cannot obtain a Machine ID for this computer.
933	The Machine ID in the Software Key does not match the machine ID of the computer it is on. Either the computer has been significantly upgraded or the Software Key file has been illegally copied to another machine.
944	The machine detected is different to the machine that the dongle was locked to.
948	Detected two different DinkeyServer programs running on the network. This is not allowed. One DinkeyServer can serve multiple products.
952	Detected USB port sharing for non-Net dongle (this setting is specified in DinkeyAdd | General | Advanced Options).
973	This dongle has been blocked by DinkeyOMS.

974	The dongle is using the DinkeyOMS blocking feature but OMSService has not contacted DinkeyOMS in the last N days for this dongle. Therefore the dongle has expired. Read the DinkeyOMS documentation for more information.

DinkeyChange Error Codes

758	Cannot open the Dinkey Update Code File (DUCF file).
759	Not a valid Dinkey Update Code file.
762	The Update Code is in an invalid format.
764	Invalid Code. If the Update Code has been entered manually then it has been entered incorrectly. If you are using an Update Code file then it is corrupt.
765	The Update Code is valid but does not match any of the dongles attached.
766	The update number of the code is higher than expected. The most likely cause is that the customer has failed to enter a previous update code that you have sent them.
767	The update number is lower than expected. The most likely cause is that the customer has already updated the dongle with this update code.
768	The Update Code has been generated with a version of DinkeyRemote that is too new for the version of DinkeyChange the user has. You should either use an older version of DinkeyRemote or get the customer to use a newer version of DinkeyChange.
769	The Update Code will result in too many programs stored in the dongle. Therefore none of the update has been applied.
770	Trying to add a program that already exists in the dongle. Therefore none of the update has been applied.
771	Not enough memory in the dongle to apply all the required changes requested. Therefore none of the update has been applied.
772	Trying to change or delete a file that does not exist in the dongle.
774	Trying to modify ‘per product’ network users to exceed the maximum allowed for that dongle model.
775	The Update Code wants to modify network users but the target dongle is not a network dongle.
778	You are trying to apply a short update code to a dongle that only allows strongly encrypted update codes.
780	Trying to add ‘per product’ network users to a dongle that uses ‘per licence’ network users.
781	Trying to add ‘per licence’ network users to a dongle that uses ‘per product’ network users.
782	Trying to modify ‘per licence’ network users to exceed the maximum allowed for that dongle model.
813	Adding so many executions to the current value the result value is too large to be stored.
822	Calling the DinkeyChange module with an invalid mask value. Use one of the constants defined.
823	Invalid filename specified for DCGetDiagnosticInfo call in the DinkeyChange module.
824	The product code mask specified is too long to be a valid product code (8 characters max).
842	The model value passed to DCDownloadDemoSoftwareKey is invalid.
843	The Product Code must be specified for the DCDownloadDemoSoftwareKey function.
1905	There is no Software Key available to download (you need to create one using DinkeyAdd).
1907	The Software Key has expired.
1910	The Software Key for this Machine ID has already been downloaded.
1921	Default dongle model was specified but there is more than one model available in the Demo Software Key Template. You need to specify a specific dongle model.
1922	The requested dongle model is not available in the Demo Software Key Template.

1923	The Demo Software Key Template has been disabled.
1924	You have run out of Demo Software Key activations. You need to purchases some more to allow your customers to download Demo Software Keys from any of your templates.

DinkeyAdd Module Error Codes

1902	The maximum number of Temporary Software Keys for this machine ID and Product Code has been reached.
1908	The Temporary Software Key you are trying to create already exists. You can specify the FLAGS_OVERWRITE_SWKEY to overwrite the existing key and not get this error.
1920	The Demo Software Key Template already exists. You can specify the FLAGS_OVERWRITE_SWKEY to overwrite the existing key and not get this error.
2100	Trying to save a DAPF file to a version that is too old (minimum is 6).
2101	Trying to save to an old DAPF file version which will cause features that are currently set to be lost (can overwrite if you set the flag FLAGS_LOSE_FEATURES in the call to WriteDAPF).
2102	DAPF handle passed is NULL. Either it has not been allocated properly or already freed.
2103	DAPF file is in an incorrect format (view error message for more information).
2104	TEMP_SWKEY_INFO / DEMO_SWKEY_INFO pointer passed is NULL.
2105	TEMP_SWKEY_INFO / DEMO_SWKEY_INFO structure is too small. Possibly not initialised “size” field correctly.
2106	The model element in the TEMP_SWKEY_INFO structure is invalid.
2107	The expiry date element in the TEMP_SWKEY_INFO / DEMO_SWKEY_INFO structure is invalid.
2109	The expiry date element in the TEMP_SWKEY_INFO / DEMO_SWKEY_INFO structure is in the past.
2110	The expiry date element in the TEMP_SWKEY_INFO structure is more than 14 days in the future.
2111	One of the input filenames is empty.
2112	One of the output filenames is empty.
2113	You cannot Shell-protect a Windows program with DinkeyAdd running in Linux or macOS.
2114	You cannot add protection to a PDF file under Mac or Linux.
2116	The output file already exists (call the same API with the FLAGS_OVERWRITE_OUTPUT flag if you want to overwrite this).
2117	The file_num parameter has an invalid value.
2119	The dongle number specified when creating a Software Key is outside the range specified in the DAPF file.
2120	A licence expiry date in the DAPF file expires before the expiry date of the Software Key (call the CreateTempSoftwareKey or CreateDemoSoftwareKeyTemplate API again with the FLAGS_EXPIRY_DONTCARE flag if you want to carry on regardless).
2121	The action parameter is invalid.
2123	You cannot protect files using the Shell Method if DinkeyAdd is running in macOS.
2125	You are trying to protect a PDF file but the DinkeyAdd.pdf.dat file is not found. It should be in the same directory as the DinkeyAdd module you are calling.
2128	The value for *max_days* in the DEMO_SWKEY_INFO structure is invalid.
2129	The value for the *models* bitmap in the DEMO_SWKEY_INFO structure is invalid.

DinkeyAddCmd Error Codes

2200	No DAPF file specified on the command line. This is a requirement for DinkeyAddCmd.
2201	Option /d used but no filename has been specified.
2202	Error creating/opening the file for the dongle number specified by the /d parameter.
2204	Option /f used but no file number specified.
2205	Option /a used but action valid is either invalid or not specified.
2206	Options /a and /e cannot be used together.
2207	Option /ei used but no filename specified.
2208	Option /eo used but no filename specified.
2209	Option /ei used and an input has been specified but no /eo output file has been specified. Either use no files or both input & output files.
2210	Option /eo used and an output has been specified but no /ei input file has been specified. Either use no files or both input & output files.

DinkeyRemote Module Error Codes

2301	Trying to save to an old DRPF file version which will cause features that are currently set to be lost (can overwrite if you set the flag FLAGS_LOSE_FEATURES in the call to WriteDRPF).
2302	DRPF handle passed is NULL. Either it has not been allocated properly or already freed.
2303	DRPF file is in an incorrect format (view error message for more information).
2306	Error writing to the DinkeyRemote logfile.
2307	The update number parameter (or in the DRPF file) is invalid.

DinkeyRemoteCmd Error Codes

2400	No DRPF file specified on the command line. This is a requirement for DinkeyRemoteCmd.
2401	Option /d used but no dongle number specified.
2402	Option /d used but dongle number is not a valid number.
2403	Option /u used but no update number specified.
2404	Option /u used but update number is not a valid number.
2405	Option /o used but there is no DUCF filename specified.

Technical Specifications

Dinkey Pro

Power consumption: 30mA (max)
Battery: None
Memory write cycles: > 1,000,000 (does not apply to Pro Lite as memory is not written)
Data retention: >10 years
Mean Time Between Failure (MTBF): > 3,000,000 hours
Storage temperature: -20°C to +85°C
Operating temperature: 0°C to 70°C
Relative Humidity Rating (no condensation): 0% to 95%

Standard Case
Dimensions: 50mm x 16mm x 8mm
Weight: 5g

Mini Case
Dimensions: 19mm x 17mm x 7mm
Weight: 2.5g

Dinkey FD Lite

Power consumption active/idle: 80mA / 40mA
Battery: None
Data retention: > 10 years
Flash memory write cycles: > 100,000
Storage temperature: -40°C to + 70°C
Operating temperature: -10°C to 55°C
Dimensions: 60mm x 18mm x 9mm
Weight: 9g

Dinkey FD Plus/Net

Power consumption active/idle: 30mA (max)
Battery: None
Memory write cycles: > 1,000,000
Data retention: > 10 years
Flash memory write cycles: > 100,000
Storage temperature: -20°C to + 85°C
Operating temperature: 0°C to 70°C
Relative Humidity Rating (no condensation): 10% to 90%

Red Case
Dimensions: 62mm x 21mm x 12mm
Weight: 10g

White Case
Dimensions: 72mm x 23mm x 10mm
Weight: 15g

Glossary of Terms

API Method

A method of protecting your software. You need to modify your code so that it calls a function in one of our protection modules.

DD_ERRCODE

An environment variable. If set to ON it will force your program to display error code if an error occurs.

Demo Software Key

A purely software-based key that allows you to create time-limited fully functional demos of your software.

DinkeyAdd

A program that adds protection to your program by locking it to one or more Dinkey dongles. It will also program Plus and Net dongles; create Software Keys and encrypt data files in conjunction with the Shell Method.

DinkeyChange

A program that accepts a code generated by DinkeyRemote and will change the protection parameters stored in Plus or Net models. It can also generate diagnostic information to a file which the user can send to you so you can see the protection parameters in all the dongles attached to their machine. It can be used to download Software Keys.

DinkeyLook

A program that searches for all the Dinkey Pro and FD dongles connected to a machine and displays their contents. Not intended for end users.

DinkeyRemote

A program that generates codes for the remote changing of protection parameters stored in a Plus or Net dongle. It is used in conjunction with DinkeyChange.

DinkeyRemote.log

A text file that contains a history of all the update codes and a summary of the changes you requested in DinkeyRemote for all the codes that you have generated.

DinkeyServer

A program that must be run on the dongle server so that a Net dongle can be accessed by other machines on the network.

dapf file

Dinkey**A**dd **P**arameter **F**ile. Stores the information that you specified in DinkeyAdd. There are two different formats: binary format (.dapf extension) and JSON format (.dapfj extension).

dlpf file

Dinkey**L**ook **P**arameter **F**ile. Generated by the user when obtaining diagnostics using DinkeyChange. You can view the parameters of all the dongles attached to their machine by opening the file with DinkeyLook.

drpf file

Dinkey**R**emote **P**arameter **F**ile. Stores the information that you specified in DinkeyRemote. There are two different formats: binary format (.drpf extension) and JSON format (.dapfj extension).

ducf file

Dinkey **U**pdate **C**ode **F**ile. Contains the update code generated by DinkeyRemote.

Dongle Number

By default, each dongle has a unique dongle number. You can order specific dongle numbers for an extra charge.

Dongle Server

A machine on the network that has the Net dongle attached to it. It can be any machine on the network. It must have the DinkeyServer program running for the dongle to work.

DRIS

Dinkey **R**untime **I**nformation **S**tructure. A structure that is used to pass information to and receive information from the dongle.

Executions

The number of times your protected program (or a feature of your program) can be executed before the executions expired error code is returned.

Features

A four-byte field that can be used to store program-specific information.

Last Date Used

The last date that the dongle was successfully called. This value is stored in the dongle and is used to stop the end user from modifying the system clock to beat the expiry date.

Max Days

The number of days (after it is first run) that your program will run before the expiry message is displayed.

Product Code

A string of up to 8 characters that distinguishes one protected product from another.

Protection Parameters

The values that are specified in DinkeyAdd and stored in the dongle. e.g. Executions, Expiry date, Max Days and Features.

SDSN

Software Developer's Serial Number. This number distinguishes your dongles from those of another software developers.

Shell Method

A method of protecting your software. This method automatically protects your software without you needing to modify your source code.

Temporary Software Key

If your customer’s dongle is damaged then you can create a Temporary Software Key to allow your protected software to run for a limited time until a replacement dongle arrives.

Type

Refers to the type of dongle. e.g. Pro or FD.

Update Number

A number stored in a Plus or Net dongle that is incremented by one each time a successful DinkeyRemote code is entered. It stops the user from entering the same update code twice

Index