



API Integration Guide for  
**JAVA**

[WWW.SOFTWARE-SECURITY-PRODUCTS.CO.UK/PROTECTIT](http://WWW.SOFTWARE-SECURITY-PRODUCTS.CO.UK/PROTECTIT)



# API Integration Guide for Java

## Definitions

**Activation Code:** A code sent to a user by email (default) or forwarded by other electronic means that enables the dongle values to be changed.

**Online Update:** An operation whereby a connection is made to the **Protect-IT** central database and the dongle is automatically updated there and then.

**Dongle 'slot':** A space on the dongle to store application critical information. There are 5 'slots' available for each type of data: *integer*, *text string*, *double* and *boolean*, in both the 'admin' and 'Application' access areas, totalling 40 'slots'.

**Expired Dongle:** A dongle that is not valid. A dongle whose expiry date is less than the current date (non-inclusive).

**Valid Dongle:** A dongle that is not expired. A dongle whose expiry date is greater than or equal to the current date (as stored on the dongle).

## Before You Begin The Integration

### Dongle Driver Installation

Ensure that the dongle driver is correctly installed by running the 'Sentinel System Driver Installer 7.5.0.exe'. A silent installation of the dongle driver can be performed by using the following command line (all on one line without line breaks):

```
"Sentinel System Driver Installer 7.5.0.exe" /s /v"/q  
ADDLOCAL=USB_Driver CONFIRMUPGRADE=TRUE"
```

Or by executing the batch file:

```
dongle_driver_silent_install.bat
```

All dongle related files can be found on the **Protect-IT CD** » *dongle\_driver directory*.

## Java Runtime Environment (JRE) & Java Standard Edition JDK

**Java Runtime Environment (JRE) 1.6** and above is required to run the **Protect-IT API**.

Please download and install the latest version, which can be found at: [www.java.com](http://www.java.com)

**Java Development Kit Standard Edition (JDK SE) 1.6** and above is required to compile your own source code. Please download and install the latest version, which can be found at: [java.sun.com](http://java.sun.com)

### Protect-IT API files

The **Protect-IT API** jar file: '*protectit.jar*' can be found on the **Protect-IT CD** » *protectit\_files* » *java*. Ensure that your source code has access to the **Protect-IT API** jar file '*protectit.jar*' by adding it to the classpath of your application.

It is assumed that the programmer has experience with the Java language and is able to write and compile code.

### Your ClientId

Before you can begin integrating, you will also need your *clientId* which will have been given to you along with your first dongle. The *clientId* needs to be included in your code for **Protect-IT** to correctly recognise your dongles.

## Quick Integration Guide

### Importing the SSPDongleManager class

You will need to use the class *com.ssp.protectit.SSPDongleManager* for the integration.

The following import statement will need to be added to each class where the *SSPDongleManager* is used:

```
import com.ssp.protectit.SSPDongleManager;
```

You will need to pass your *clientId* (given to you with your dongles) when you construct a new instance of *SSPDongleManager*:

```
// ClientId 1234  
SSPDongleManager manager = new SSPDongleManager(1234);
```

**Note:** You can also pass the *applicationId* for your application (if you changed it from the default of 0) and a *JFrame* instance for your application, although this is optional.

```
SSPDongleManager manager = new SSPDongleManager(1234,1);
or
JFrame frame = new JFrame(...);
SSPDongleManager manager = new SSPDongleManager(1234,0,frame);
```

### Selecting and Using a Plugged-in Dongle

In order to start using the program a valid dongle (a dongle that has not expired) must be plugged-in. If the dongle that is plugged-in has expired (or is about to expire) the user will be given the opportunity to apply an *Activation Code* or perform an *Online Update*.

```
// Selects a valid dongle to use
manager.selectValidDongle();
```

If a valid dongle is not found (even after applying an *Activation Code* or *Online Update*) the program will not be started and the user has the option to exit at this stage, otherwise, normal program operation continues.

### Dongle Data

The dongle is capable of storing various types of data.

**Protect-IT** allows you to save the following data types:

- **String** – Text of no more than 32 characters.
- **Integer** – A whole (positive or negative) number between 0 and 65,535.
- **Double** – A (positive or negative) number to the accuracy of 3 decimal places between 0 and 4,294,967.295.
- **Boolean** – Stores a value of either true (on) or false (off).

There are 5 ‘slots’ on the dongle for each of the above data types in both the ‘Admin’ and ‘Application’ access areas.

### Admin Access Area

This is an area of the dongle that you (as the *Administrator*) have full control of and should be used for application configuration, enabling/disabling modules, etc.

The *Admin Access Area* values **can not** be modified programmatically. *Admin Access Area* values can only be updated through *Activation Codes* and *Online Updates*.

### Application Access Area

This is an area of the dongle that your application has full control of and should be used for data that is relevant to the working of the application but which you (as an *Administrator*) do not have direct access to.

It could be used for storing user settings for your application, application counters, etc.

*Online Updates* and *Activation Codes* **do not** update the *Application Access Area*.

### Accessing Your Dongle Data

You have the ability to programmatically read and write different types of data. The table below describes how each bit of data can be used programmatically or by the application of an *Activation Code* (AC) or conducting an *Online Update* (OU).

Property Name	Programmatically		AC/OU	Description
	Read	Write	Write	
Client Id	✓	✗	✗	Your unique <i>ClientId</i> sent to you with your dongles.
(Optional) Application Id	✓	✗	✓	(Optional) use it if you have more than one application to protect.
(Dongle) Serial	✓	✗	✗	The unique dongle id.
Expiry Date	✓	✗	✓	Controls the expiry mechanism for your application.
(Optional) Username	✓	✗	✓	(Optional) use it to give a meaningful name to each dongle. E.g. <i>Client 1</i> , <i>Client 2</i> .
Language Code	✓	✗	✓	A 2 characters-long code (ISO-639-1) used for the automatic display of internationalised messages to the user (can also be used internally by your application).
(Optional) Country Code	✓	✗	✓	(Optional) A 2 characters-long code (ISO-3166) used for the automatic display of internationalised messages to the user (can also be used internally by your application).

TABLE CONTINUES ON NEXT PAGE

Property Name	Programmatically		AC/OU	Description
	Read	Write	Write	
Integer 1–5	✓	✗	✓	5 'slots' on the dongle for storing whole (positive or negative) numbers between 0 and 65,535. ( <i>Admin Access Area</i> )
String 1–5	✓	✗	✓	5 'slots' on the dongle for storing text of no more than 32 characters. ( <i>Admin Access Area</i> )
Double 1–5	✓	✗	✓	5 'slots' on the dongle for storing (positive or negative) numbers to the accuracy of 3 decimal places between 0 and 4,294,967.295. ( <i>Admin Access Area</i> )
Boolean 1–5	✓	✗	✓	5 'slots' on the dongle for storing values of either true (on) or false (off). ( <i>Admin Access Area</i> )
Application Integer 1–5	✓	✓	✗	5 'slots' on the dongle for storing whole (positive or negative) numbers between 0 and 65,535. ( <i>Application Access Area</i> )
Application String 1–5	✓	✓	✗	5 'slots' on the dongle for storing text of no more than 32 characters. ( <i>Application Access Area</i> )
Application Double 1–5	✓	✓	✗	5 'slots' on the dongle for storing (positive or negative) numbers to the accuracy of 3 decimal places between 0 and 4,294,967.295. ( <i>Application Access Area</i> )
Application Boolean 1–5	✓	✓	✗	5 'slots' on the dongle for storing values of either true (on) or false (off). ( <i>Application Access Area</i> )

```
// Programmatically Reading/Writing dongle Data
int clientId = manager.readClientId();

int applicationId = manager.readApplicationId();

int serial = manager.readSerial();

// The Expiry Date in "yyyymmdd" format
String expiryDate = manager.readExpiryDate();

String username = manager.readUsername();

// 2 characters-long
String languageCode = manager.readLanguageCode();

// 2 characters-long or empty string ""
String countryCode = manager.readCountryCode();

// Admin Access Area
int integer1 = manager.readInteger1();

// Application Access Area
manager.writeApplicationInteger1(12345);

int appInteger1 = manager.readApplicationInteger1();

...

int integer5 = manager.readInteger5();

manager.writeApplicationInteger5(-200);

int appInteger5 = manager.readApplicationInteger5();

// Admin Access Area
String string1 = manager.readString1();

// Application Access Area
manager.writeApplicationString1("Hello");

String appString1 = manager.readApplicationString1();

...
```

```
String string5 = manager.readString5();
manager.writeApplicationString5("World");
String appString5 = manager.readApplicationString5();
// Admin Access Area
double double1 = manager.readDouble1();
// Application Access Area
manager.writeApplicationDouble1(12.5);
double appDouble1 = manager.readApplicationDouble1();

double double5 = manager.readDouble5();
manager.writeApplicationDouble5(12.345);
double appDouble5 = manager.readApplicationDouble5();
// Admin Access Area
boolean boolean1 = manager.readBoolean1();
// Application Access Area
manager.writeApplicationBoolean1(true);
boolean appBoolean1 = manager.readApplicationBoolean1();
...
boolean boolean5 = manager.readBoolean5();
manager.writeApplicationBoolean5(false);
boolean appBoolean5 = manager.readApplicationBoolean5();
```

Activation Codes/Online Updates can be distributed/set from **Protect-IT**'s central online database accessible via any internet enabled browser.

## Handling Errors

All error handling is automated. If there is a problem then the appropriate message will be displayed on screen where the user can choose an appropriate action.

## Handling Application Exit

The default behaviour for **Protect-IT** is to just exit the application *System.exit(1)* when either:

- the user clicks on an available 'Exit Application' button (from the dongle manager and dongle expired dialog boxes);
- the user has not properly installed the dongle driver;
- the user failed to plug-in a dongle (or removed one whilst the application was running);
- the user modified his system clock (Time Cheating).

This behaviour can be overridden by implementing the interface:

`com.ssp.protectit.SSPExitHandler`

The method `exit` has to be implemented, as in the example below:

```
import com.ssp.protectit.SSPExitHandler;
public class MyExitHandler implements SSPExitHandler {
public void exit( int status, String lastMethodCall ) {
// You could decide your action based on the status
switch (status) {
// Dongle driver not properly installed or dongle never inserted
// on system for it to be recognised
case SSPDongleManager.DONGLE_LIBRARY_NOT_FOUND: ...;
// The dongle is not plugged-in.
case SSPDongleManager.DONGLE_NOT_FOUND: ...;
```

```

// A dongle that was being used has been removed
case SSPDongleManager.DONGLE_SPECIFIC_DONGLE_NOT_FOUND: ...;

// User clicked exit application on dongle Expired window
case SSPDongleManager.DONGLE_EXPIRED: ...;

// User modified the System clock (Time Cheating)
case SSPDongleManager.TIME_CHEAT: ...;

// User clicked exit application on the dongle Manager window
case SSPDongleManager.NO_ERROR:...;
}

// You can use the Exit Handler to finish any tasks that may have
// been started, such as: Closing opened stream/resources, saving
// files presenting extra options like 'Are you sure you want to
// exit?' dialog boxes, etc.

// You could also base your decision on the last method called
if ( lastMethodCall.equals("readSerial") ) {
System.out.println( "Failed at readSerial." );
System.exit(1);
}

// You must remember to call System.exit as it is not
// automatically called when using an ExitHandler.
System.exit(1); // Exits the application
}
}

```

After creating and compiling your class that implements *SSPExitHandler* you must register it with the *SSPDongleManager* instance, please ensure this is done before the *selectValidDongle* method is called:

```

SSPDongleManager manager = new SSPDongleManager(1234);
manager.setExitHandler( new MyExitHandler() );
manager.selectValidDongle();

```

### Maintaining the Security of Your Application

The **Protect-IT API** protects your application and ensures revenue by:

- only allowing your application to run on a valid (non expired dongle), giving you control over the expiry date. E.g. Only update your application's monthly/quarterly *Expiry Date* once rental fees have been received;
- strong encryption that makes it extremely difficult to extract your information, clone the dongle, record and repeat dongle communication;
- employing a '*time cheating*' mechanism that will instantly disable your application if a user turns back the clock in order to get more use (once the user changes his clock back to normal, the application will resume its normal operation).

In order to ensure that all of the above works to give your application (and revenue) the best protection, the following guidelines should be followed:

1. Only send a new *Activation Code* (authorise *Online Update*) to extend the *Expiry Date* after all payments have been received. From our experience, updating on a monthly basis has been very effective;
2. Store any critical information to the running of your application on the dongle '*slots*' provided (integer, string, double, boolean). This ensures that only with a valid dongle can your application be used and also gives you absolute control on all those critical values (i.e. they can be modified by *Activation Code* or *Online Update* along with the monthly expiry date update). This is another good reason to keep the expiry dates monthly, as it forces users to not only update the expiry date but all the other dongle settings as well;

3. Call the protect method at various important stages of your application running (and sporadically, if possible) to ensure that the correct dongle is still plugged-in and hasn't been removed to start the application up on another computer.

```
// Call this method many times during the running of your
// application, at every critical operation and sporadically
// through its normal use.
manager.protect();
```

## Extras

You can call the *Activation Code Window* or the *Online Update Window* in code by:

```
manager.showActivationCodeWindow();
manager.showOnlineUpdateWindow();
```

By default the local system language is used to set the language to be used for the **Protect-IT API**. You can set the language used by calling the setLocale("language code", "country code†"):

```
// Your ClientId
SSPDongleManager manager = new SSPDongleManager(1234);

// Sets the language to English
manager.setLocale("en","");
manager.selectValidDongle();
```

The following languages are currently supported, with many more on their way: "en" English, "es" Spanish, "it" Italian, "de" German, "pt" Portuguese.

\* The language code is a 2 characters-long text string representing a language as listed in ISO-639-1.

† The country code is a 2 characters-long text string representing a country as listed in ISO-3166.

## Quick Reference

Sample use of the **Protect-IT API**. (Also refer to the *samples* directory on the CD)

```
package my.package;

import com.ssp.protectit.SSPDongleManager;

public class Main {
    protected SSPDongleManager manager;

    public Main() {
        // Initialise the dongle Manager
        manager = new SSPDongleManager(1234); // Your ClientId
        manager.selectValidDongle();
        startMyApplication();
    }

    public void randomMethod() {
        ...
        // Protection call (ensures dongle is valid)
        manager.protect();
        ...
        // Let user apply an Activation Code (Optional, as automatically
        // handled when dongle expires)
        manager.showActivationCodeWindow();
    }

    public int doSomething() {
        // E.g. need the value of Admin Integer 1 and Application Integer
        // 2 for a calculation
```





**Software Security Products Limited**

The Manor House, 260 Ecclesall Road South, Sheffield, South Yorkshire, S11 9PS, United Kingdom

**t** +44(0) 1142 217 070 **f** +44(0) 1142 218 080 **e** [support@software-security-products.co.uk](mailto:support@software-security-products.co.uk)

**w** [www.software-security-products.co.uk](http://www.software-security-products.co.uk)