

# RED-App Technical User Guide

Prepared For



## Table of Contents

1. Prerequisites .....	3
2. Package Content .....	3
3. Environment Setup .....	4
4. Project compilation and output.....	4
5. RED-App .....	6
6. RED-Demo .....	7
7. REDApp client code development guidelines.....	9

## 1. Prerequisites

- JDK, preferable latest one – v.1.8.0.60 for Windows x86  
<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>
- Visual Studio 2013
- Ant, preferable latest one – v.1.9.6  
<http://apache.mivzakim.net/ant/binaries/apache-ant-1.9.6-bin.zip>
- Sentinel LDK License manager v.6.31 or newer
- Stratasys Sentinel Master Key
- Stratasys Sentinel HL Max Driverless Key, firmware 4.51

## 2. Package Content

### **proj**

#### **firmware**

firmware\_update\_wle\_451.v2c – Apply this file on the key to upgrade the key firmware to 4.51

#### **bin** – tools folder

##### **windows**

vmbuilder.exe - App On Chip (AoC) SDK tool which allows to compile Java source code into AoC compatible encrypted image and install it on a dongle.

licgen\_tmpl\_windows.exe – LicGen command line utility

asn1dump\_windows.exe – utility to print the structure of an ASN.1 data object

Technical User Guide

**openssl** – A library for creating key pairs and signing messages

**cunit** – A library for C Unit tests

**red** – sources and output folder

build.xml – Ant build script

CMakeLists.txt – CMake build script (optional, can be used to create Visual Studio solution etc.)

**src** – sources folder

hasp\_api.h – HASP API (AoC SDK) definitions

hasp\_io\_buffer.c - HASP API helper functions

hasp\_io\_buffer.h - HASP API helper functions definitions

libhasp\_windows\_demo.lib – DEMOMA HASP API library with AoC support

libhasp\_windows\_92199.lib – Stratasys HASP API library with AoC support

hasp\_vm\_api.c – vm functions (init invoke and close)

**red\_demo.c** – main RED-Demo source file. Stratasys developers should examine this file.

REDApp\_utils.c – utility functions used by RED-demo, wrappers of Java functions

REDApp\_utils.h – definitions of utility functions and Java wrappers

**app** – Java sources

**REDApp.java** – RED-App source file. This file should be edited by Stratasys

developers and four Application Placeholders (CNOF, CEOP, CNOP, and CYSO) should be implemented.

#### **token**

##### **classes**

Crypto.java – Crypto functions

Except.java - AoC library function definitions

Sys.java - AoC library function definitions

**out** – output folder, will be created after running build.xml in Ant

red\_demo.exe – RED-Demo executable

### 3. Environment Setup

- Install JDK, by default will be installed to C:\Program Files (x86)\Java\jdk1.8.0\_60.
- Set JAVA\_HOME environment variable to C:\Program Files (x86)\Java\jdk1.8.0\_60.
- Download and unzip Ant, for example to C:\Tools\apache-ant-1.9.6-bin\apache-ant-1.9.6.
- Set ANT\_HOME environment variable – for example - C:\Tools\apache-ant-1.9.6-bin\apache-ant-1.9.6.
- Add to PATH environment variable following: %ANT\_HOME%\bin.
- Restart machine.

### 4. Project compilation and output

Ant script (build.xml) executes following tasks:

- 3.1 Compiles REDApp Java sources with JDK into class files, class files will appear in proj/red/out/app folder.
- 3.2 Invokes vmbulder.exe utility which converts Java class files into AoC image and transfers encrypted image into connected Sentinel HL Driverless Key.  
**Please note:** Stratasys Sentinel Master Key and Sentinel HL Driverless Key should be connected to the machine.
- 3.3 Compiles C sources with Visual Studio C Compiler into proj/red/out/red\_demo.exe.

#### Steps

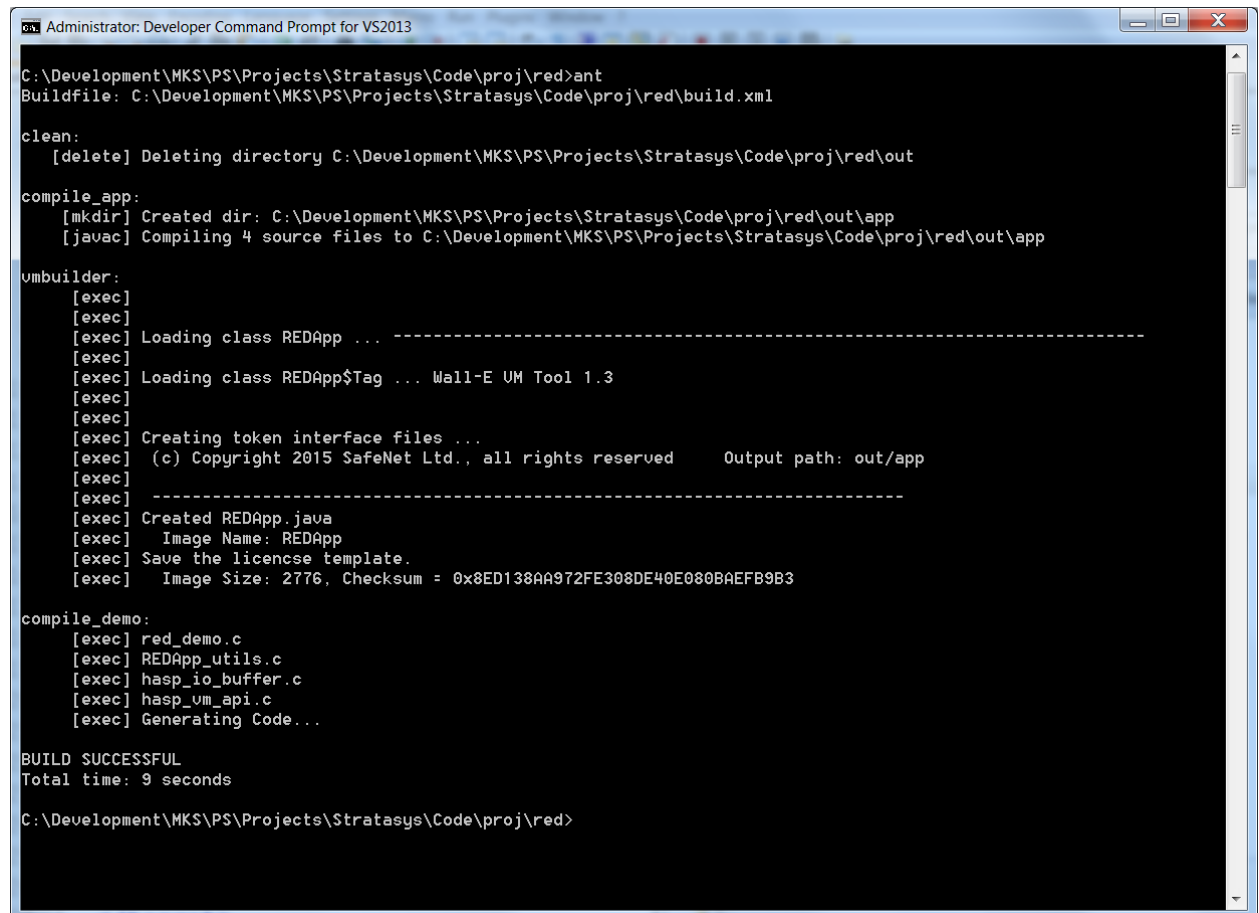
1. Update firmware of your keys by applying **proj\firmware\Firmware\_update\_451.v2c** file in Sentinel Admin Control Center.
2. Open Visual Studio Command Prompt: Start->All Programs-> Visual Studio 2013->Visual Studio Tools-> Developer Command Prompt for VS2013.
3. Change current directory to proj/red.

4. Type: **ant** and hit enter.

Ant script (build.xml) executes following tasks:

- 3.1 Compiles REDApp Java sources (using JDK) into class files, class files will appear in proj/red/out/app folder.
- 3.2 Converts compiled Java classes into AoC encrypted image and transfer it into a dongle.
- 3.3 Compiles C sources with Visual Studio C Compiler into proj/red/out/red\_demo.exe.

Normal output will look like this:



```
Administrator: Developer Command Prompt for VS2013

C:\Development\MKS\PS\Projects\Stratasys\Code\proj\red>ant
Buildfile: C:\Development\MKS\PS\Projects\Stratasys\Code\proj\red\build.xml

clean:
  [delete] Deleting directory C:\Development\MKS\PS\Projects\Stratasys\Code\proj\red\out

compile_app:
  [mkdir] Created dir: C:\Development\MKS\PS\Projects\Stratasys\Code\proj\red\out\app
  [javac] Compiling 4 source files to C:\Development\MKS\PS\Projects\Stratasys\Code\proj\red\out\app

umbuilder:
  [exec]
  [exec]
  [exec] Loading class REDApp ... -----
  [exec]
  [exec] Loading class REDApp$Tag ... Wall-E UM Tool 1.3
  [exec]
  [exec]
  [exec] Creating token interface files ...
  [exec] (c) Copyright 2015 SafeNet Ltd., all rights reserved    Output path: out/app
  [exec]
  [exec] -----
  [exec] Created REDApp.java
  [exec] Image Name: REDApp
  [exec] Save the licence template.
  [exec] Image Size: 2776, Checksum = 0x8ED138AA972FE308DE40E080BAEFB9B3

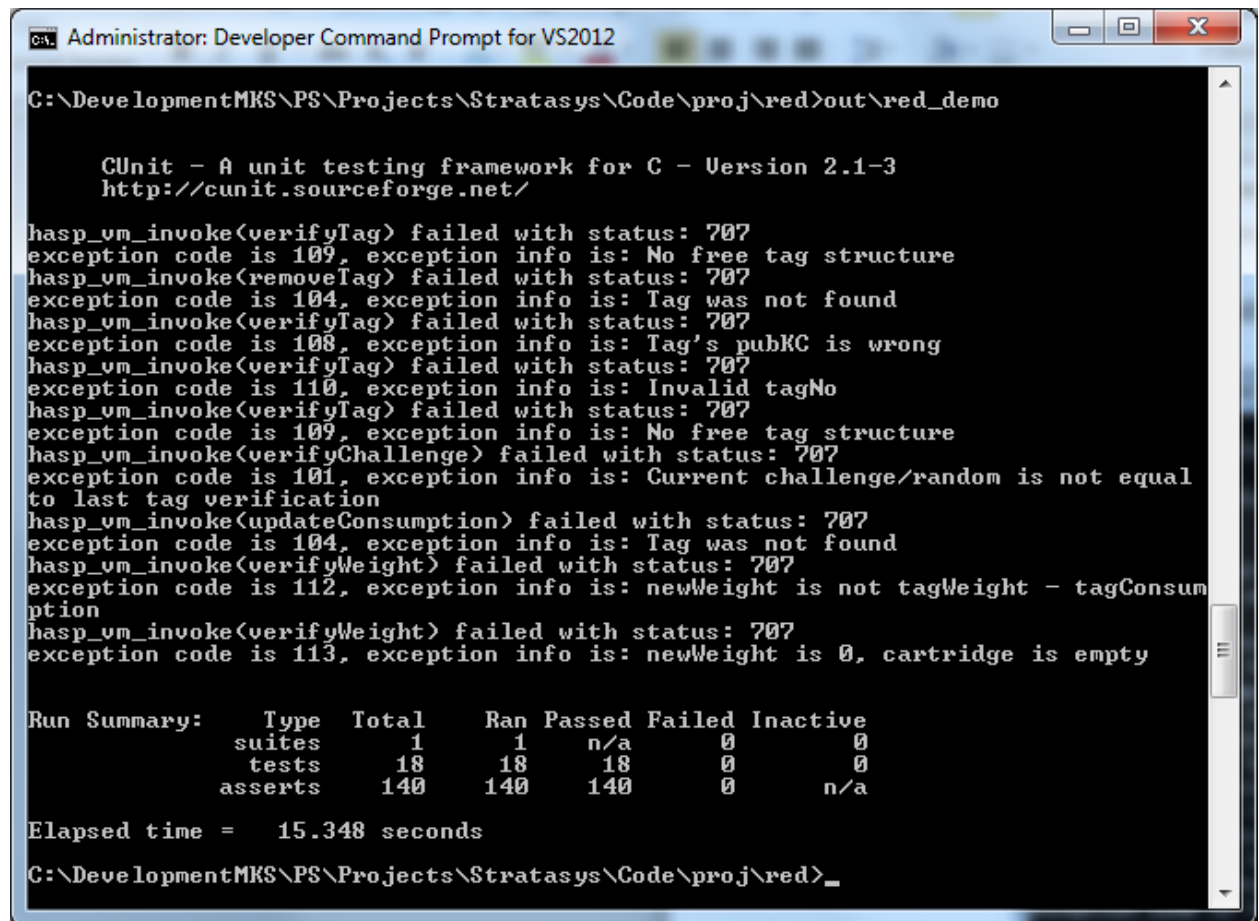
compile_demo:
  [exec] red_demo.c
  [exec] REDApp_utils.c
  [exec] hasp_io_buffer.c
  [exec] hasp_um_api.c
  [exec] Generating Code...

BUILD SUCCESSFUL
Total time: 9 seconds

C:\Development\MKS\PS\Projects\Stratasys\Code\proj\red>
```

5. Now red\_demo.exe, which contains wrappers of REDApp methods and some test cases (unit test macros) which use these methods, can be executed in Windows Command Line utility and run on the key.

Normal output will look like this:



```
Administrator: Developer Command Prompt for VS2012

C:\DevelopmentMKS\PS\Projects\Stratasys\Code\proj\red>out\red_demo

CUUnit - A unit testing framework for C - Version 2.1-3
http://cunit.sourceforge.net/

hasp_vm_invoke(<verifyTag>) failed with status: 707
exception code is 109, exception info is: No free tag structure
hasp_vm_invoke(<removeTag>) failed with status: 707
exception code is 104, exception info is: Tag was not found
hasp_vm_invoke(<verifyTag>) failed with status: 707
exception code is 108, exception info is: Tag's pubKC is wrong
hasp_vm_invoke(<verifyTag>) failed with status: 707
exception code is 110, exception info is: Invalid tagNo
hasp_vm_invoke(<verifyTag>) failed with status: 707
exception code is 109, exception info is: No free tag structure
hasp_vm_invoke(<verifyChallenge>) failed with status: 707
exception code is 101, exception info is: Current challenge/random is not equal
to last tag verification
hasp_vm_invoke(<updateConsumption>) failed with status: 707
exception code is 104, exception info is: Tag was not found
hasp_vm_invoke(<verifyWeight>) failed with status: 707
exception code is 112, exception info is: newWeight is not tagWeight - tagConsum
ption
hasp_vm_invoke(<verifyWeight>) failed with status: 707
exception code is 113, exception info is: newWeight is 0, cartridge is empty

Run Summary:
  Type    Total    Ran    Passed    Failed    Inactive
  suites      1      1      n/a      0      0
  tests     18     18      18      0      0
  asserts   140    140    140      0      n/a

Elapsed time = 15.348 seconds

C:\DevelopmentMKS\PS\Projects\Stratasys\Code\proj\red>_
```

## 5. RED-App

RED-App source file REDApp.java is located in proj\red\src\app.

This file contains implementation of file methods, as required by SRS:

1. **public static boolean verifyTag(byte tagNo, boolean isActive, byte[] certificate, byte[] random)**  
tagNo, isActive, certificate – input parameters  
certificate – consists of  
Cartridge and tag info (35 bytes)  
Cartridge public key (72 bytes),  
signature length (2 bytes),  
signature (72 bytes) – 181 bytes all together. Signature length bytes contain value 72.  
random – output parameter (however byte array of length 8 should be initialized in the client code)

2. **public static boolean verifyChallenge(byte tagNo, boolean isActive, byte[] signedChallenge)**  
tagNo, isActive, signedChallenge – input parameters  
signedChallenge contains the challenge (8 bytes at offset 8) –  
signature length (2 bytes offset 32),  
signature of the first 32 bytes (72 bytes offset 34) 106 bytes all together
3. **public static boolean removeTag(byte tagNo)**  
tagNo – input parameter
4. **public static boolean updateConsumption(byte tagNo, boolean isActive, int consumption, byte[] random)**  
tagNo, isActive, consumption – input parameters  
random – output parameter (however byte array of length 8 should be initialized in the client code)
5. **public static boolean verifyWeight(byte tagNo, boolean isActive, byte[] signedWeight)**  
tagNo, isActive, signedWeight – input parameters  
signedWeight – consists of:  
Tag random (8 bytes)  
Tag serial number (8 bytes)  
RED random (8 bytes)  
New Weight (4 bytes),  
signature length (2 bytes),  
signature (72 bytes) – 102 bytes all together. Signature length bytes contain value 72.

All above functions return a boolean value in case of functional negative flow (for example, a signature cannot be verified) and also in case of error (for example – certificate byte array cannot be parsed, or its length is invalid).

In case an exception is thrown from a function, the return value in the invoker will be the exception number and the exception message is returned in the iobuffer of invoker.

In addition, REDApp class provides placeholders for four methods, which should be implemented by Stratasys developers:

6. **public static int CNOF(int SW, int SR, int LHO1200, int SRX, int IEF)**
7. **public static int CEOP(int SOP, int SW, int SR, int LHO1200, int AF, int IEF)**
8. **public static int CNOP(int SH, int SST, int SYO, int HPW, int SPEO, int CH)**
9. **public static int CYSO(int SH, int SST, int SYO, int HPW, int NGIP, int NOP, int SPEO, int YINOP, int YSINOP)**

**Please note:** Above methods should be invoked through AoC API in the C client code, this is explained later.

## 6. RED-Demo

RED-Demo sources are located in proj\red\src folder.

There are libhasp\_windows\_92199.lib, few C header and C source files, and 2 libraries: openssl and cunit (excluding app folder, which contains RED-App sources) - see above, in the Package Content chapter.

All above files should be included into Stratasys software sources, excluding red\_demo.c, openssl and cunit libraries, which contains test cases of Java methods wrappers.

**REDApp\_utils.h** and **REDApp\_utils.c** provide wrappers for all appropriate Java methods:

1. `hasp_status_t verifyTag(hasp_handle_t handle, byte tagNo, bool isActive, const byte *certificate, unsigned char certificateLen, byte *random, int *result);`
  2. `hasp_status_t verifyChallenge(hasp_handle_t handle, byte tagNo, boolean isActive, byte *signedChallenge, unsigned char signedChallengeLen, int *result);`
  3. `hasp_status_t removeTag(hasp_handle_t handle, byte tagNo, int *result);`
  4. `hasp_status_t updateConsumption(hasp_handle_t handle, byte tagNo, bool isActive, int consumption, byte *random, bool *result);`
  5. `hasp_status_t verifyWeight(hasp_handle_t handle, byte tagNo, boolean isActive, byte *signedWeight, unsigned char signedWeightLen, bool *result);`
  6. `hasp_status_t CNOF(hasp_handle_t handle, int SW, int SR, int LH01200, int SRX, int IEF, int *result);`
  7. `hasp_status_t CEOP(hasp_handle_t handle, int SOP, int SW, int SR, int LH01200, int AF, int IEF, int *result);`
  8. `hasp_status_t CNOP(hasp_handle_t handle, int SH, int SST, int SYO, int HPW, int SPEO, int CH, int *result);`
  9. `hasp_status_t CYSO(hasp_handle_t handle, int SH, int SST, int SYO, int HPW, int NGIP, int NOP, int SPEO, int YINOP, int YSINOP, int *result);`
- Please refer to above Java definitions to learn about input/output parameters and format of certificate, signedChallenge and signedWeight.
  - Result parameter holds 1 for success 0 for failure.
  - In case of debug mode (in the Java code) result parameter will hold exception number.

In addition, **REDApp\_utils.h** and **REDApp\_utils.c** provide two basic AoC service methods:

1. `hasp_status_t openAoC(hasp_handle_t *handle)`  
Must be called in the beginning, logs in to HASP key and initializes AoC VM engine.  
Handle is output parameter and should be saved in the client code and used in all further calls to AoC functions (above 9 functions) as well as to below closeAoC function.  
If a dongle was disconnected, the session will end. In this case, the handle must be closed (using closeAoC), and the openAoC function must be called again. Note that the dongle will also lose the REDApp state, which means that the initialization procedure has to be done again.
2. `void closeAoC(hasp_handle_t handle)`  
This ends the communication with HASP by closing the VM and HASP API handles.

**red\_demo.c** demonstrates usage of the Java wrappers, provides test cases functions which organizes calls to wrappers in various scenarios:



- *Test case 1 - Successful initialization:*  
16 (verifyTag) + 16 (verifyChallenge) calls of authenticateTag with isActive = false, followed by 8 (verifyTag) + 8 (verifyChallenge) calls with isActive = true.
- *Test case 2 – testing removeTag:*
  1. Adding additional tag when all slots are occupied.
  2. Removing a tag.
  3. Removing a tag that doesn't exist
  4. Adding a tag successfully after a tag was removed.
  5. Updating a tag.
- *Test case 3 - Failure scenarios for verifyTag*
  1. Verification is not passed
  2. Certificate cannot be parsed or signature length is invalid
  3. tagNo is invalid (less than min or greater than max)
  4. tagNo is valid, but tag cannot be found
- *Test case 4 - Failure scenarios for verifyChallenge*
  1. Verification is not passed
  2. tagNo != lastTagNo or isActive != lastIsActive
- *Test case 5 (added in Milestone 3) - success scenarios for verifyWeight*
  1. First update weight (Consumption 0)
  2. Update weight (Consumption != 0)
- *Test case 6 (added in Milestone 3) - Failure scenarios for verifyWeight*
  1. random is not equal to lastRandom
  2. New weight is not equal to subtraction of consumption from weight
  3. New weigh is 0

## 7. REDApp client code development guidelines

- Please use **proj\firmware\Firmware\_update\_451.v2c** to update keys firmware to new one – 4.51.
- Include in your application following files:
  - a. proj\red\src\libhasp\_windows\_92199.lib
  - b. proj\red\src\hasp\_api.h
  - c. proj\red\src\hasp\_vm\_api.c
  - d. proj\red\src\hasp\_io\_buffer.c
  - e. proj\red\src\hasp\_io\_buffer.h
  - f. proj\red\src\REDApp\_utils.c
  - g. proj\red\src\REDApp\_utils.h
- Call openAoC at the beginning and closeAoC at the end, as shown in the RED Demo Suite\_1 init function.
- Use hasp\_handle\_t handle, output of openAoC, in all further calls to function wrappers.

