RED-App Technical User Guide

Prepared For

Stratasys®

Table of Contents

Important: This document should be used for referring to Milestone 1 of project 'Enhanced Sentinel HL-Stratasys'. In Milestone 1, low level crypto services - verifyMessage and generateRandom don't call real crypto methods in Firmware, they are just mock implementations.

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 2010 or newer
- 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 Time Driverless Key

2. Package Content

proj

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.

VMBuilderView - App On Chip (AoC) SDK GUI tool which allows to convert compiled Java classes into AoC compatible encrypted image and install it on a dongle

VMBuilderView_windows.exe – VMBuilderView executable.

cmd Install

haspdint.exe - Sentinel LDK License manager setup, v.6.62

readme.html - Sentinel LDK License manager setup help

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.)

doc – docs folder

Technical Guide.docx

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

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

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.

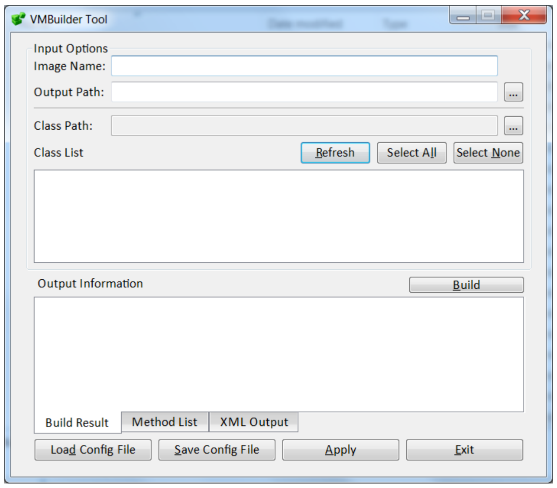
3.2 Invokes vmbulder.exe utility which converts Java class files into AoC image and transfers encrypted image into connected Sentinel HL Driverless Key.
Important note:
In Milestone 1, the use of the vmbuilder.exe utility is disabled in the build.xml file, because it only supports DEMOMA dongles. This limitation of the vmbuilder.exe utility will be resolved in Milestone 2 only. To create the AoC image and program it into the connected Sentinel HL Driverless Key, the VMBuilderView.exe GUI utility has to be used. The tool is not scriptable, which means that this conversion and programming is a simple but manual step (select Java class files, specify output folder, select non-DEMOMA mode, program the AoC image into the connected Stratasys dongle).

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. Open Visual Studio Command Prompt (Start->All Programs->Visual Studio 2010->Visual Studio Tools-> Visual Studio Command Prompt (2010)).
2. Change current directory to proj/red.
3. 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 Compiles C sources with Visual Studio C Compiler into proj/red/out/red_demo.exe.
4. Convert compiled Java classes into AoC encrypted image and transfer it into a dongle
   4.1 Open VMBuilderView_windows.exe from proj\bin\windows\VMBuilderView



4.2 Click on ‘Load Config File’ and load configuration file from proj\red\ REDApp.vmp:

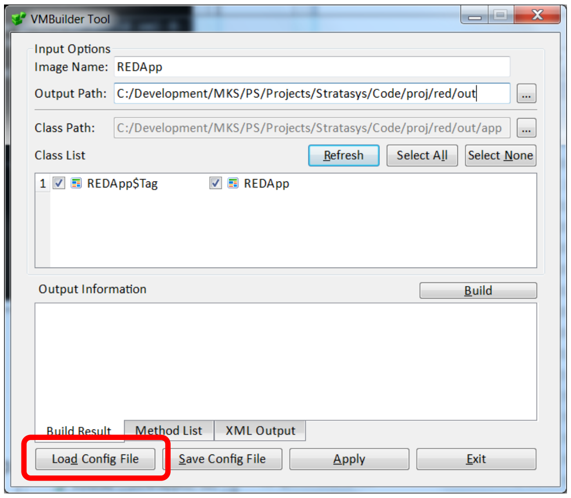


Image Name: REDApp
Output Path: C:/Development/MKS/PS/Projects/Stratasys/Code/proj/red/out
Class Path: C:/Development/MKS/PS/Projects/Stratasys/Code/proj/red/out/app
Class List
1 ☑ REDApp$Tag ☑ REDApp

4.3 Click on Build and build AoC image, examine Build Result to see that the build was successful:

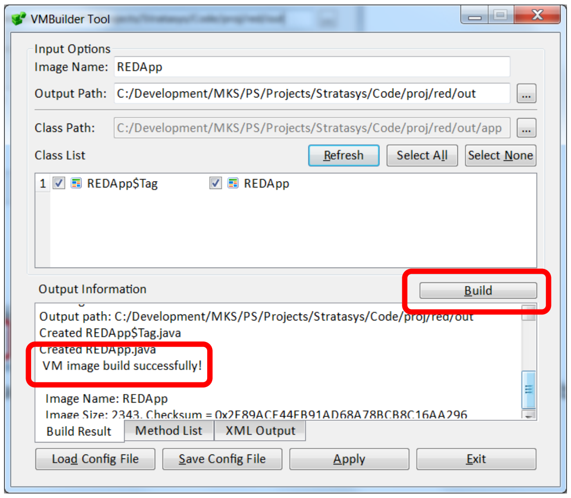
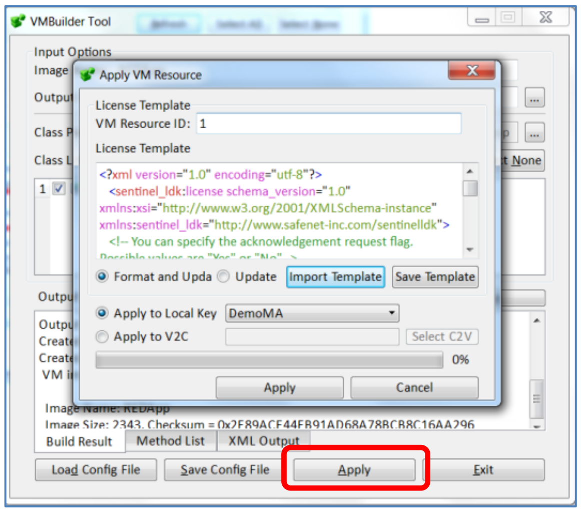


Image Name: REDApp
Output Path: C:/Development/MKS/PS/Projects/Stratasys/Code/proj/red/out
Class Path: C:/Development/MKS/PS/Projects/Stratasys/Code/proj/red/out/app
Class List
1 ☑ REDApp$Tag ☑ REDApp

```
Output path: C:/Development/MKS/PS/Projects/Stratasys/Code/proj/red/out
Created REDApp$Tag.java
Created REDApp.java
 VM image build successfully!

  Image Name: REDApp
  Image Size: 2343. Checksum = 0x2F89ACF44FB91AD68A78BCB8C16AA296
```

4.4 Click on Apply to open ‘Apply VM Recourse’ dialog:



Apply VM Resource

License Template
VM Resource ID: 1
License Template

```
<?xml version="1.0" encoding="utf-8"?>
  <sentinel_ldk:license schema_version="1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:sentinel_ldk="http://www.safenet-inc.com/sentinelldk">
  <!-- You can specify the acknowledgement request flag.
```

Format and Upda · Update · Import Template · Save Template

Apply to Local Key: DemoMA

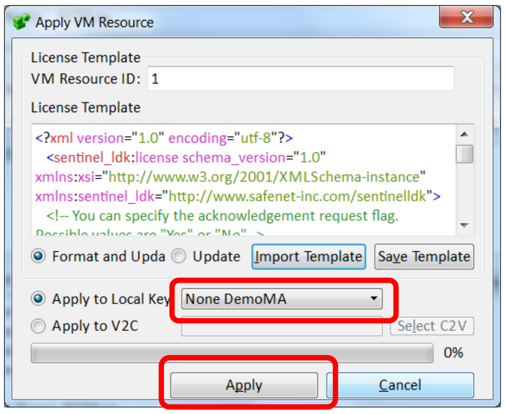
Apply to V2C · Select C2V

0%

Apply · Cancel

```
Image Name: REDApp
Image Size: 2343, Checksum = 0x2F89ACF44FB91AD68A78BCB8C16AA296
```

4.5 Change ‘Apply to Local Key’ drop down to ‘Non DemoMA’ and click Apply:



Apply VM Resource

License Template
VM Resource ID: 1
License Template

```
<?xml version="1.0" encoding="utf-8"?>
  <sentinel_ldk:license schema_version="1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:sentinel_ldk="http://www.safenet-inc.com/sentinelldk">
  <!-- You can specify the acknowledgement request flag.
```

Format and Upda · Update · Import Template · Save Template

Apply to Local Key: None DemoMA

Apply to V2C · Select C2V

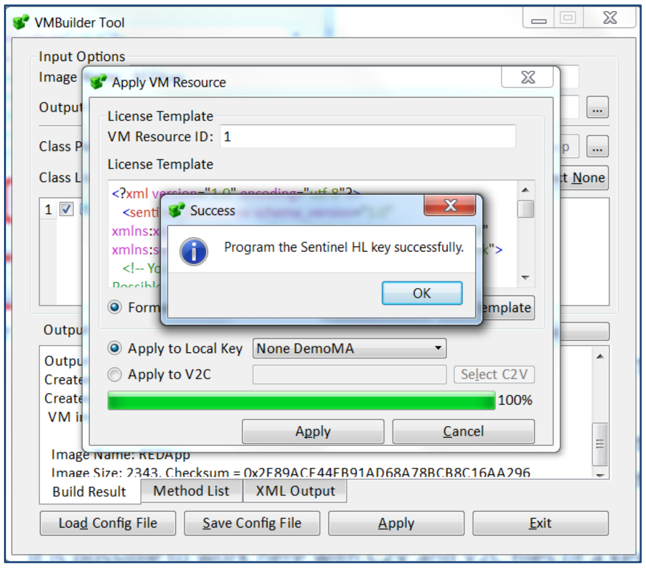
0%

Apply · Cancel

Please note:

1. Stratasys Sentinel Master Key and Stratasys Sentinel HL Time (or Max) Driverless Key should be connected.
2. As soon as ‘Format and Update’ radio button is chosen, the key will be formatted first and then AoC image will be transferred to the key. If you want not to format a key, choose ‘Update’ radio button.
3. It is possible to work here with C2V and V2C files (if a key is disconnected) – Mater key should be connected still!

4.6 Now the key is programmed with REDApp:


VM Resource ID: 1

Program the Sentinel HL key successfully.

Apply to Local Key: None DemoMA

100%

Image Name: REDApp

Image Size: 2343, Checksum = 0x2F89ACF44FB91AD68A78BCB8C16AA296

4.7 Now red_demo.exe, which contains wrappers of REDApp methods and some use cases (scenario functions) which use these methods, can be executed in Windows Command Line utility and run on the key.

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 message (72 bytes), signature length (2 bytes), signature (72 bytes) – 146 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[] signature)**
   tagNo, isActive, signature – input parameters
   signature – 72 bytes

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 weight (4 bytes), random (8 bytes), signature length (2 bytes), signature (72 bytes) – 86 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).
Please note: in Milestone 2 there will be possibility to report specific error to the C client code in debug mode (REDApp.java could be compiled then id Debug or Release mode), this will be explained in Technical Guide for Milestone 2 document.

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 (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, which contains test cases (scenarios) 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, bool *result);`

2. `hasp_status_t verifyChallenge(hasp_handle_t handle, byte tagNo, boolean isActive, byte *signature, unsigned char signatureLen, bool *result);`

3. `hasp_status_t removeTag(hasp_handle_t handle, byte tagNo, bool *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 LHO1200, int SRX, int IEF, int *result);`

7. `hasp_status_t CEOP(hasp_handle_t handle, int SOP, int SW, int SR, int LHO1200, 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 and signedWeight.

In addition, REDApp_utils.h and REDApp_utils.c provide two basic AoC service methods:

1. `hasp_status_topenAoC(hasp_handle_t *handle)`
   Must be called in the beginning, logins 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 utility functions which organizes calls to wrappers together to well defined procedures, like authenicateTag in various scenarios:

➢ *Scenario 1 - Successful initialization:*
16 (verifyTag) + 16 (verifyChallenge) calls of authenticateTag with isActive = false, followed by 8 (verifyTag) + 8 (verifyChallenge) calls with isActive = true.

➢ *Scenario 2 – testing removeTag:*
   1. Adding additional tag when all slots are occupied - should fail.
   2. Removing a tag.
   3. Removing a tag that doesn't exist - should fail
   4. Adding a tag successfully after a tag was removed.
   5. Updating a tag.

➢ *Scenario 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

➢ *Scenario 4 - Failure scenarios for verifyChallange*
   1. Verification is not passed
   2. tagNo != lastTagNo or isActive != lastIsActive

- *Scenario 5: Successful Scenarios for updateConsumption/verifyWeight pair*
    1. First update weight (Consumption 0)
    2. Update weight (Consumption != 0)

- *Scenario 6: 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.

- *Scenario 7 – Testing Application Placeholders*
    1. CNOF
    2. CEOP
    3. CNOP
    4. CYSO

All above scenario functions are called from main procedure.

7. REDApp client code development guidelines

- 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_io_buffer.c
    d. proj\red\src\hasp_io_buffer.h
    e. proj\red\src\REDApp_utils.c
    f. proj\red\src\REDApp_utils.h
- Call openAoC at the beginning and closeAoC at the end, as shown in main procedure.
- Use hasp_handle_t handle, output of openAoC, in all further calls to function wrappers.