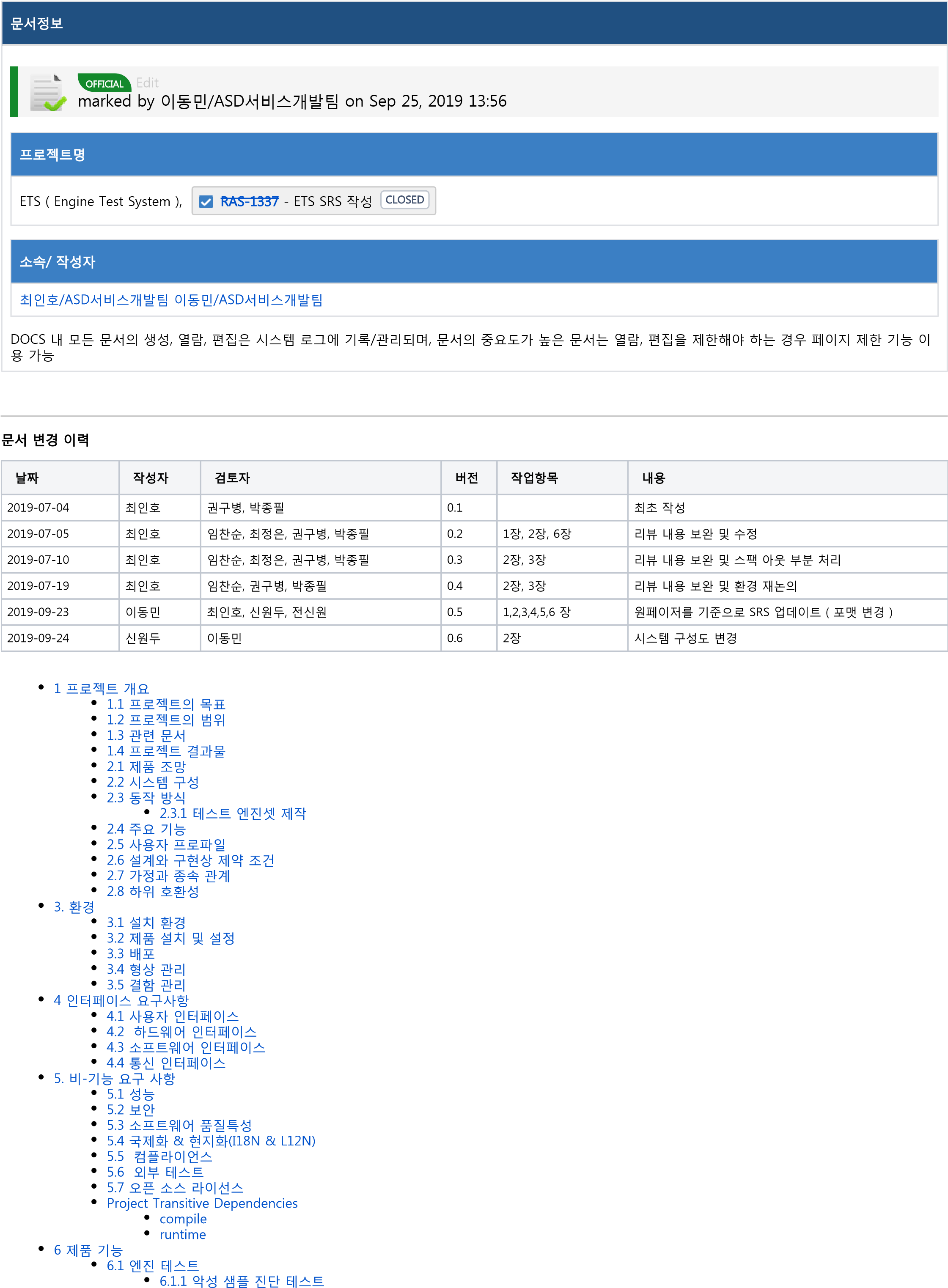
ETS ( Engine Test System ) SRS



6.1.1.1 엔진셋 복사

6.1.1.2 샘플셋 확인

6.1.1.3 샘플셋 복사

6.1.1.4 샘플셋 진단

6.1.1.6 결과 처리 6.1.2.1 제품 업데이트 서버 설정

# 6.1.2 엔진 업데이트 테스트

6.1.2.2 제품 업데이트 실행

6.1.2.3 제품 실시간 검사 상태 확인

6.1.2.4 제품 엔진 Version Check 6.1.2.1 Mupdate 업데이트 테스트

# 6.1.2 엔진 업데이트 테스트

## 6.1.2.1.1 환경 설정

6.1.2.1.1.1 가상의 CDN 환경을 구성한다.

6.1.2.1.1.2 Mupdate 은 아래와 같이 셋팅한다.

6.1.2.1.1.2 Mupdate 을 실행한다.

## 6.1.2.2 Autoup 업데이트 테스트

### 6.1.2.2.1 환경 설정

6.1.2.2.1.1 V3 제품 설치 후 자동 업데이트 기능 OFF

6.1.2.2.1.2 Silence 모드의 실행압축파일을 생성한다.

6.1.3.2.1.3 Command 로 실행압축 파일을 실행한다.

6.1.3.2.3 제품의 엔진버전을 확인한다.

6.1.2.3 엔진 파일 무결성 확인

6.1.3.4 엔진 초기화

6.1.3.5 실행 결과 확인

6.1.3.6 결과 처리 6.1.2.1 빌드 유형 체크

# 6.1.3 실시간 검사 테스트

6.2

테스트 인프라

6.3

가상화 환경 컨트롤

6.1.2.2 엔진셋 복사

6.1.2.3 유저(User) 모드 실행

6.1.2.4 커널(Kenel) 모드 실행

6.1.2.5 실행 결과 확인

6.1.2.6 결과 처리

# 6.1.3 엔진 모듈 초기화 TBD

6.1.3.1 제품별로 설정된 엔진모듈 현황을 정리한다.

6.1.3.2 ahn.ui 파일에서 제품별 파일 그룹을 정리한다.

6.1.3.3 엔진 패치셋에서 제품별 엔진 파일을 취합하여 구성한다.

6.1.3.4 test\_asc를 통해 엔진 초기화 및 로딩을 해본다.

# 6.2.1 테스트 환경 설정

6.2.1.1 개발 및 TEST 목적의 RAS Server 구축 및 활용

6.2.1.2 개발 및 TEST 목적의 Oasis Storage 구축 및 활용 ( epack & NEA )

6.2.1.3 개발 및 TEST 목적의 WEB - WAS Server 구축 및 활용

6.2.1.4 개발 및 TEST 목적의 Jenkins Master - Agent 구축 및 활용

# 6.3.1 가상 이미지 관리

6.3.1.1 개발기간 가상 이미지 관리

6.3.1.2 가상 이미지 운영 전환 전략

6.3.1.3 개발 가상이미지 구성 Set 과 운영 가상이미지 구성 Set 관리 방안

# 6.4 엔진 업데이트 환경 설정

6.4.1 가상 업데이트 서버 관리

6.4.2 업데이트 모듈 설정 6.5.1 테스트 서버 상태 확인 6.6.1. V3 제품 컨트롤

# 6.5 모니터링 6.6. 기타

6.6.2. RAS 시스템 전체 구성도 ( ETS 표시됨 . publish by 신원두/ASD서비스개발팀 )

1. 대응 시스템 구성도

# 1 프로젝트 개요

## 1.1 프로젝트의 목표

실시간으로 배포되는 엔진에 대한 테스트를 담당하여 악성 샘플에 대한 진단값 누락없이 안전한 엔진 패치셋이 제공되도록 하는 것을 목표로 한다.

* ETS는 Engine TEST 를 뜻한다.
* 10년이 넘은 엔진 테스트 환경에 대해서 renewal을 통해 보다 안전한 엔진이 배포될 수 있도록 테스트를 강화한다.- 테스트 환경에 대한 지속적인 모니터링 강화 및 상황 지표(metric)를 추가하여 안정적으로 운영될 수 있도록 한다.

## 1.2 프로젝트의 범위

본 프로젝트는 RAS를 통해 생성되는 엔진 패치셋에 대한 테스트 내용을 다룬다.

* RAS 6.0 환경을 기반으로 하는 새로운 엔진 테스트 환경을 구성하여 개발한다.
* TS 엔진의 binary 영역의 테스트는 제외하며 data 영역에 대한 테스트만 진행한다.
* 소프트웨어 QA팀에서의 요구사항을 반영한다.
* 엔진모듈별 초기화 및 로딩 테스트- 테스트는 아래 범위로 진행한다.
* 샘플 진단 테스트
* 엔진 패치셋 업데이트 테스트
* Window 제품 한정 테스트
* V3 ES 9.0
* 비 Windows 제품에 대한 테스트는 추후에 진행한다.

NEXT

## 1.3 관련 문서

구분

URL

Onepager

[ETS ( Engine Test System ) Onepage](https://docs.ahnlab.com/display/RAS/ETS+%28+Engine+Test+System+%29+Onepager)

[r](https://docs.ahnlab.com/display/RAS/ETS+%28+Engine+Test+System+%29+Onepager)

시나리오

[ETEST 테스트 시나리](https://docs.ahnlab.com/x/PQAODQ)

[오](https://docs.ahnlab.com/x/PQAODQ)

비지니스 아키텍쳐

[https://docs.ahnlab.com/x/2HnXD](https://docs.ahnlab.com/x/2HnXDQ)

[Q](https://docs.ahnlab.com/x/2HnXDQ)

어플리케이션 아키텍쳐

[https://docs.ahnlab.com/x/43nXD](https://docs.ahnlab.com/x/43nXDQ)

[Q](https://docs.ahnlab.com/x/43nXDQ)

데이터 아키텍쳐

[https://docs.ahnlab.com/x/5nnXD](https://docs.ahnlab.com/x/5nnXDQ)

[Q](https://docs.ahnlab.com/x/5nnXDQ)

시스템 아키텍쳐

[https://docs.ahnlab.com/x/dEnXD](https://docs.ahnlab.com/x/dEnXDQ)

[Q](https://docs.ahnlab.com/x/dEnXDQ)

## 1.4 프로젝트 결과물

* 엔진 테스트 인프라

구분 서버타입 대수 비고

Web Server - Apache ( CentOS ) 가상 1 RND 망

Was Server - Tomcat ( CentOS ) 가상 1 RND 망

Test Master Server - Jenkins ( CentOS ) 가상 RND 망

Test Host PC - Agent ( Windows ) - 엔진 테스트용 가상 1 샘플망

Test Host PC - Agent ( Windows ) - 패치셋 업데이트 테스트용 가상 1 RND 망

Test Host PC - Agent ( Windows ) - SFX 업데이트 테스트용 가상 1 RND 망

* 빌드 및 테스트환경
* ETS 빌드 및 ALM

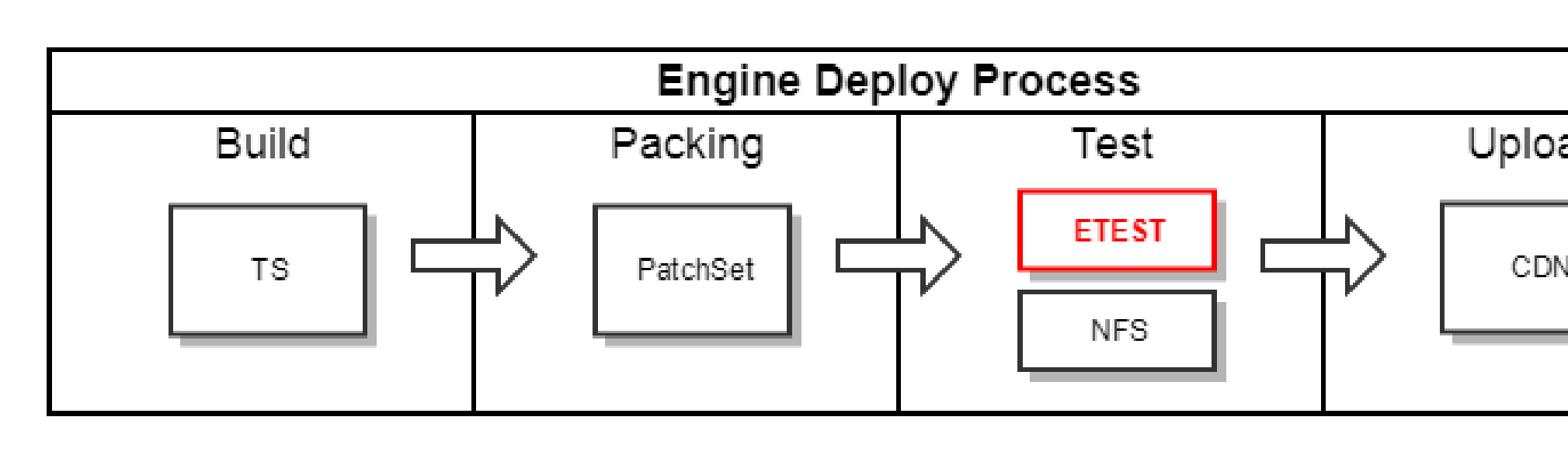
ETS 는 WORKS ( project code : RAS ) - DOCS ( space code : RAS ) - REPOS ( project code : RAS ) BUILDS ( project code : RAS ) - ABIS ( grdoup name : com.ahnlab.asd , artifact name : ets.war ) 의 안랩 개발 플랫폼을 사용하도록 하며, Maven 빌드 체계를 활용한다.

* ETS 테스트환경테스트 환경은 운영환경의 RAS 구성을 동일하게 복제한 테스트환경을 구축하여 테스트를 진행한다.

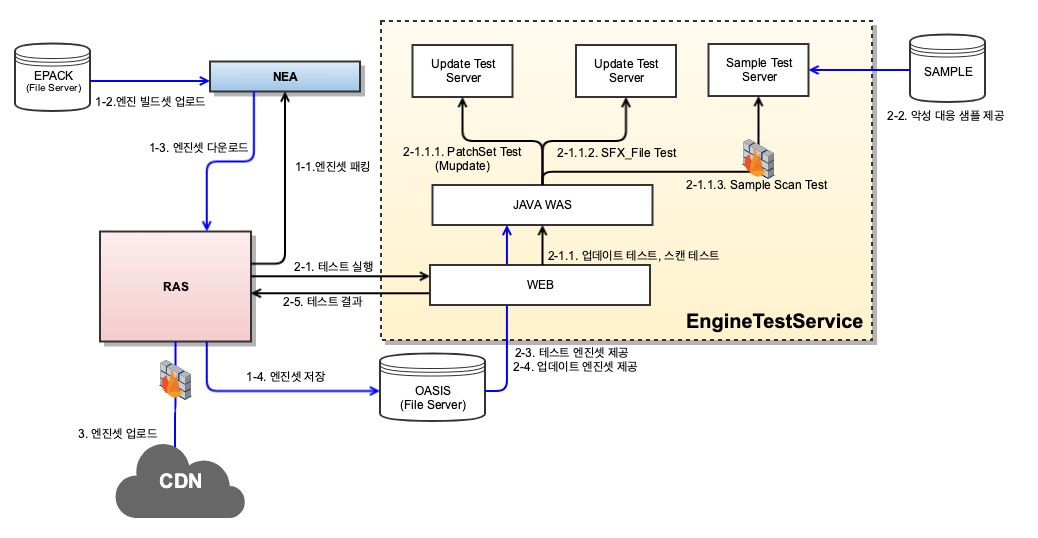
## 2.1 제품 조망

엔진 배포 과정중 아래와 같이 Test 영역의 일부로 운영된다.

ETS : 악성 샘플 진단(실시간 포함) 및 업데이트 테스트를 담당한다.



## 2.2 시스템 구성



구분

설명

RAS

Release Automation Service로 엔진 및 제품 패치셋에 대한 자동 배포를 지원하는 서비스다.

NEA

NextEasyAhn 서비스로 패치셋을 생성을 위해 파일 및 패치셋 정보를 관리하는 서비스다.

CDN

패치셋을 올리는 곳으로 제품의 패치 업데이트를 지원한다.

ETS

엔진 패치셋에 대한 새로운 테스트 환경을 말한다.

Update Test Server : 엔진 업데이트 테스트를 주관하는 서버다.

Sample Test Server : 악성 샘플에 대한 진단 테스트를 주관하는 서버다.

EPACK

엔진 빌드셋이 저장되는 파일저장소다.

엔진 패킹시 NEA서비스로 파일이 업로드 된다.

OASIS

RAS를 통해 생성된 엔진 패치셋들에 대한 저장소다.

TCL 수행시 엔진 패치셋을 ETS에서 가져갈 수 있도록 한다.

SAMPLE

TS 엔진 빌드시 엔진 버전별 대응된 샘플이 저장되는 저장소다.

분석가에 의해 수동으로 진단값이 추가된 샘플에 대해서만 존재한다.

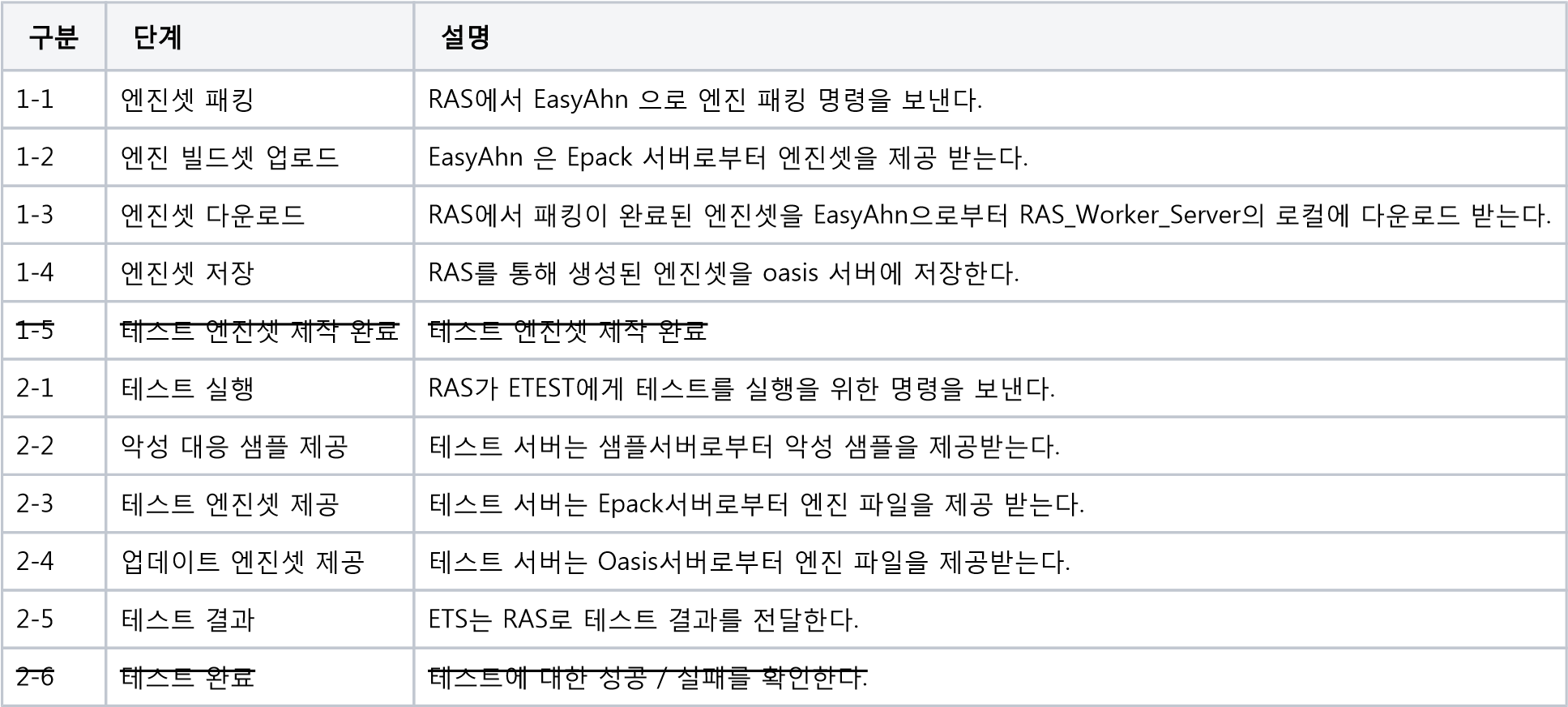
해당 서버로는 readonly 권한으로 접근하며,

이 있다.

분석팀에 ownership

## 2.3 동작 방식

2.3.1 테스트 엔진셋 제작



3 엔진셋 업로드 RAS에서는 테스트가 완료되면 CDN쪽으로 엔진셋을 업로드 한다.

## 2.4 주요 기능

시스템은 다음과 같은 기능을 제공한다.

# 제목 상세 내용

6.1 엔진 테스트 엔진 테스트 기능을 제공한다.

6.1.1 악성 샘플 진단 테스트 TS 엔진으로 악성 샘플에 대한 진단 테스트를 제공한다.

~~6.1.2 실시간 검사 테스트 TS 엔진의 실시간 검사 드라이버의 정상 동작 테스트를 제공한다.~~

6.1.3 엔진 업데이트 테스트 TS 엔진에 대해서 mupdate, autoup 등의 업데이트 테스트를 제공한다.

6.1.4 엔진 모듈 초기화 엔진 모듈에 대해서 제품별 테스트를 제공한다. NEXT

~~6.2 테스트 인프라 엔진 테스트에 관련된 인프라쪽 관리를 제공한다.~~

~~6.2.1 엔진 업데이트 환경 설정 mupdate를 통해 업데이트 할 수 있는 가상의 업데이트 환경을 제공한다.~~

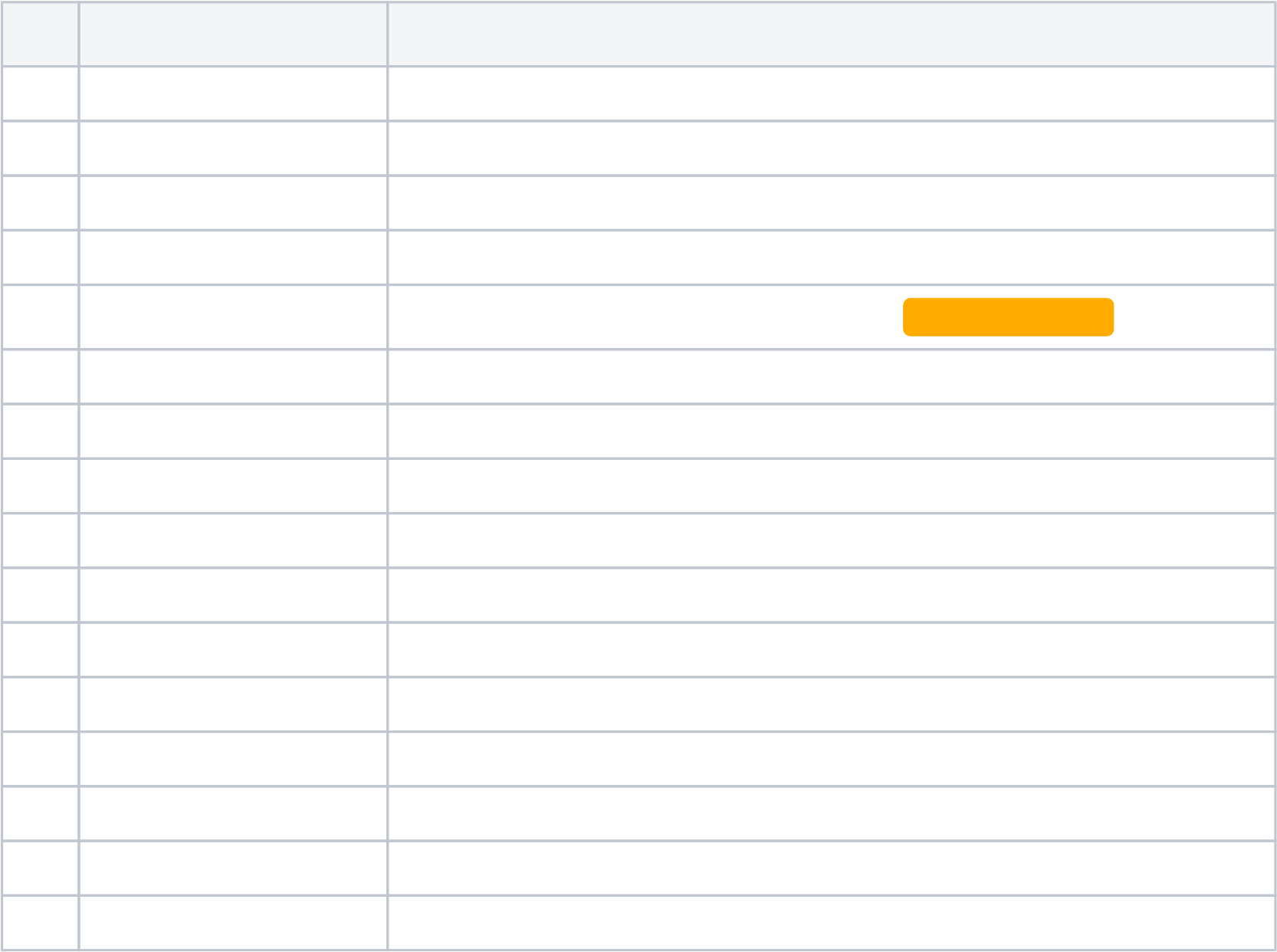
~~6.2.2 테스트 서버 이미지 관리 테스트 이미지에 대해서 시스템 명령 혹은 스냅샷 관리를 제공한다.~~

~~6.3 가상화 환경 컨트롤 테스트 인프라 가상화 환경을 컨트롤 할 수 있는 기능을 제공한다.~~

~~6.3.1 가상 이미지 관리 가상 이미지를 관리(재부팅, Revert, 스냅샥) 기능을 제공한다.~~

~~6.4 엔진 업데이트 환경 설정 엔진 업데이트 테스트를 위한 환경 셋팅 기능을 제공한다.~~

~~6.4.1 가상 업데이트 서버 관리 임시의 CDN 서비스를 제공한다.~~

 ~~6.4.2 업데이트 모듈 설정 테스트를 위한 mupdate, autoup 으로 엔진 업데이트가 가능하도록 한다. (우4)~~

6.5 모니터링 ETEST 인프라와 테스트 진행에 대한 정보를 제공한다.

6.5.1 테스트 서버 상태 테스트 서버의 상태를 보여준다.

6.5.2 테스트 진행 상태 테스트 진행 상태를 보여준다.

2.5 사용자 프로파일

N/A

## 2.6 설계와 구현상 제약 조건

아래의 제약 조건이 존재한다.

* ~~악성 샘플을 핸들링 할 수 있도록 격리된 가상화 환경으로 악성 샘플 테스트를 진행한다.~~
* 테스트 환경은 TCL에 따라 환경이 다르다.
* 샘플 진단을 위한 테스트 환경은 아래와 같다.
* 네크워크망 : 샘플망
* V3 제품 설치 여부 : 미설치
* 엔진 업데이트에 관한 테스트 환경은 아래와 같다.
* 네트워크망 : RND망
* V3 제품 설치 여부 : 설치
* 테스트 마스터 서버와 통신하는 Agent 결과가 RAS 마스터에 결과를 Noti하며 TCL이 진행되어야 한다.
* 샘플망에 대한 방화벽 적용이 가능하여야 한다.
* TCL에 제품이 추가될 경우 테스트 설정 파일로 컨트롤하여 확장할 수 있도록 한다.
  1. 가정과 종속 관계

TS엔진이 binary와 data로 분리되어 빌드할 수 있는 환경을 가정으로 한다.

* 1. 하위 호환성

하위 호환성은 고려하지 않는다.

# 3. 환경

## 3.1 설치 환경

모든 테스트는 OS의 아키텍처로 구분되기에 특정 OS 선정 후 모든 아키텍처를 지원하는 설치 환경으로 지원한다.

구분 내용 ( Server )

OS CentOS 7 Minimal Stable Version 총 3대

Java OpenJRE 1.8.181 or Oracle Jre 1.8.181

Server Apache Web Server, Tomcat Was Server, Jenkins Master Server

구분 내용 ( Test Agent )

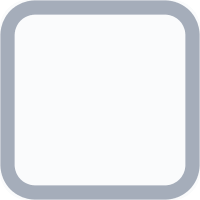
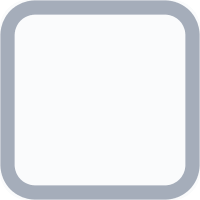
OS Windows10 (x64) 총 3대 - Test Host

V3 AhnLab V3 Endpoint Security 9.0 - 9.0.56.1(Build 1418)

(정기점검 시점에 제품 버젼업그레이드를 검토)

Java OpenJRE 1.8.181 or Oracle Jre 1.8.181 Agent Jenkins Remote Agent

## 3.2 제품 설치 및 설정



아래와 같은 제품 설치 및 설정이 필요하다.

구분

샘플 진단 테스트 환경

엔진 업데이트 테스트 환경

서버 유형

가상

가상

V3 제품 설치

미설치

V3 ES 9.0

V3제품

옵션

-

업데이트 방법

자동 업데이트 사용(권장) → 해당 기능 OFF

업데이트 고급 설정

업데이트할 때 V3패치 파일도 다운로드 → 해당 기능 OFF

(

기타 옵션은 V3제품의 기본값으로 운영한다

)

네트워크

SMPL 샘플서버망

(

정적 분석망

)

RND망

## 3.3 배포

ETS ( Engine Test System )의 Middleware 설치본인 WAR 확장자 Artifact는 ABIS에 배포하며, Maven 빌드 스크립트에 의하여 빌드 머쉰에서 WAS 로 Auto Hot Deploy를 진행합니다.

## 3.4 형상 관리

분류

정보

이슈관리

[https://works.ahnlab.com/projects/RA](https://works.ahnlab.com/projects/RAS)

[S](https://works.ahnlab.com/projects/RAS)

( Component : ETS ) 관리

문서관리

[https://docs.ahnlab.com/display/RA](https://docs.ahnlab.com/display/RAS)

[S](https://docs.ahnlab.com/display/RAS)

( TAG : ETS ) 관리

형상관리

[https://repos.ahnlab.com/projects](https://repos.ahnlab.com/projects/AIM)

[/](https://repos.ahnlab.com/projects/AIM)

RAS

빌드관리

[https://builds.ahnlab.com/browse/RA](https://builds.ahnlab.com/browse/RAS)

[S](https://builds.ahnlab.com/browse/RAS)

산출물관리

스냅샷 관리

릴리즈 관리

[http://abis.ahnlab.com/artifactory/list/maven-local-repos-snapshots/com/ahnlab/asd](http://abis.ahnlab.com/artifactory/list/maven-local-repos-snapshots/com/ahnlab/asd/)

[/](http://abis.ahnlab.com/artifactory/list/maven-local-repos-snapshots/com/ahnlab/asd/)

h

[ttp://abis.ahnlab.com/artifactory/list/maven-local-repos-releases/com/ahnlab/asd](http://abis.ahnlab.com/artifactory/list/maven-local-repos-releases/com/ahnlab/asd/)

[/](http://abis.ahnlab.com/artifactory/list/maven-local-repos-releases/com/ahnlab/asd/)

3.5 결함 관리

Works를 사용하여 이슈 관리를 수행한다.

# 4 인터페이스 요구사항

## 4.1 사용자 인터페이스

N/A - 본 프로젝트는 RAS 의 Module System 으로써 RAS에 의하여 동작되며, RAS에 결과를 회신하는 방식으로 사용자 인터페이스를 따로이 제공하고 있지 않다. ( Middle ware 기준 )

단. Jenkins Master user interface는 기 제공되고있는 화면을 그대로 사용하는 전제를 가진다.

## 4.2 하드웨어 인터페이스

N/A - 본 프로젝트는 최소 하드웨어 요구사항을 제한할 이유가 없다.

이유는 Vmware 가상화 시스템을 활용하여 최소 5개 이상의 시스템 구성을 할 예정이며, 각 Layer 마다 Spec을 ScaleUp 할 수 있기 때문이다. 관련 하드웨어 스펙은 1.4 장에 적시하고 있다.

## 4.3 소프트웨어 인터페이스

RAS - Middleware Java Application 간 Communication interface는 Rest API를 사용하도록 한다. return json.

Middleware - Jenkins Master Server 간 Communication interface는 Middleware에서 Jenkins Client jar 라이브러리를 사용하여 통신하도록 한다.

Jenkins Master - Jenkins Remote Host 간의 Communication interface는 JMX를 활용하여 통신하도록 한다.

또한 본 프로젝트는 자바 8 버전 이상의 JRE 가 요구된다. ( Oracle 및 OpenJDK 포함 ) 또한 Oracle Database를 사용하고있다.

## 4.4 통신 인터페이스

Sample Test 목적의 HOST는 Sample 망에서 동작하므로,

Jenkins 마스터와의 JMX 통신을 위하여 필요하다면 방화벽 신청을 통해 통신하도록 한다.

# 5. 비-기능 요구 사항

5.1 성능 현재 운영중인 ETEST에서 테스트 소요시간을 기준하여 짧은 시간내에 테스트를 완료하여야 한다.

테스트 종류 지표 목표 비고

샘플 테스트 소요시간 30초 현재 운영중인 ETEST 에서의 소요시간 ~~실시간 테스트 소요시간 90초 현재 운영중인 ETEST 에서의 소요시간~~ 업데이트 테스트 소요시간 420초 현재 운영중인 ETEST 에서의 소요시간 추가로, 엔진 패치셋에 대한 패킹 작업 완료 전에 모든 테스트가 완료되어야 한다.

대응 유형 패킹 소요시간 테스트 셋 생성 소요시간 테스트 허용 최대 시간

수시 20분 5분 내외 15분 정기 30분 10분 내외 20분

5.2 보안 생략 5.3 소프트웨어 품질특성

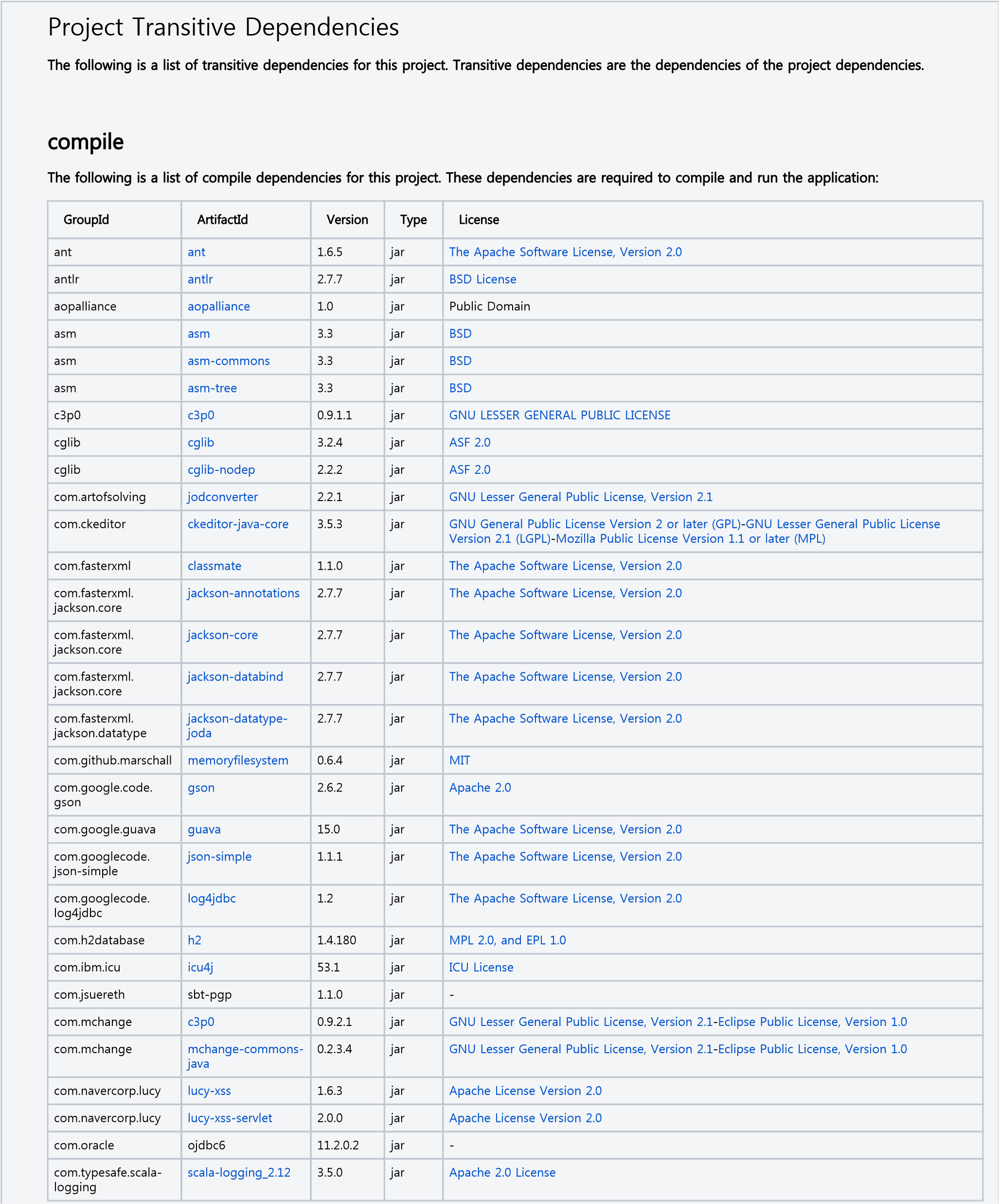
생략

5.4 국제화 & 현지화(I18N & L12N)

생략

## 5.5 컴플라이언스 생략 5.6 외부 테스트 생략 5.7 오픈 소스 라이선스

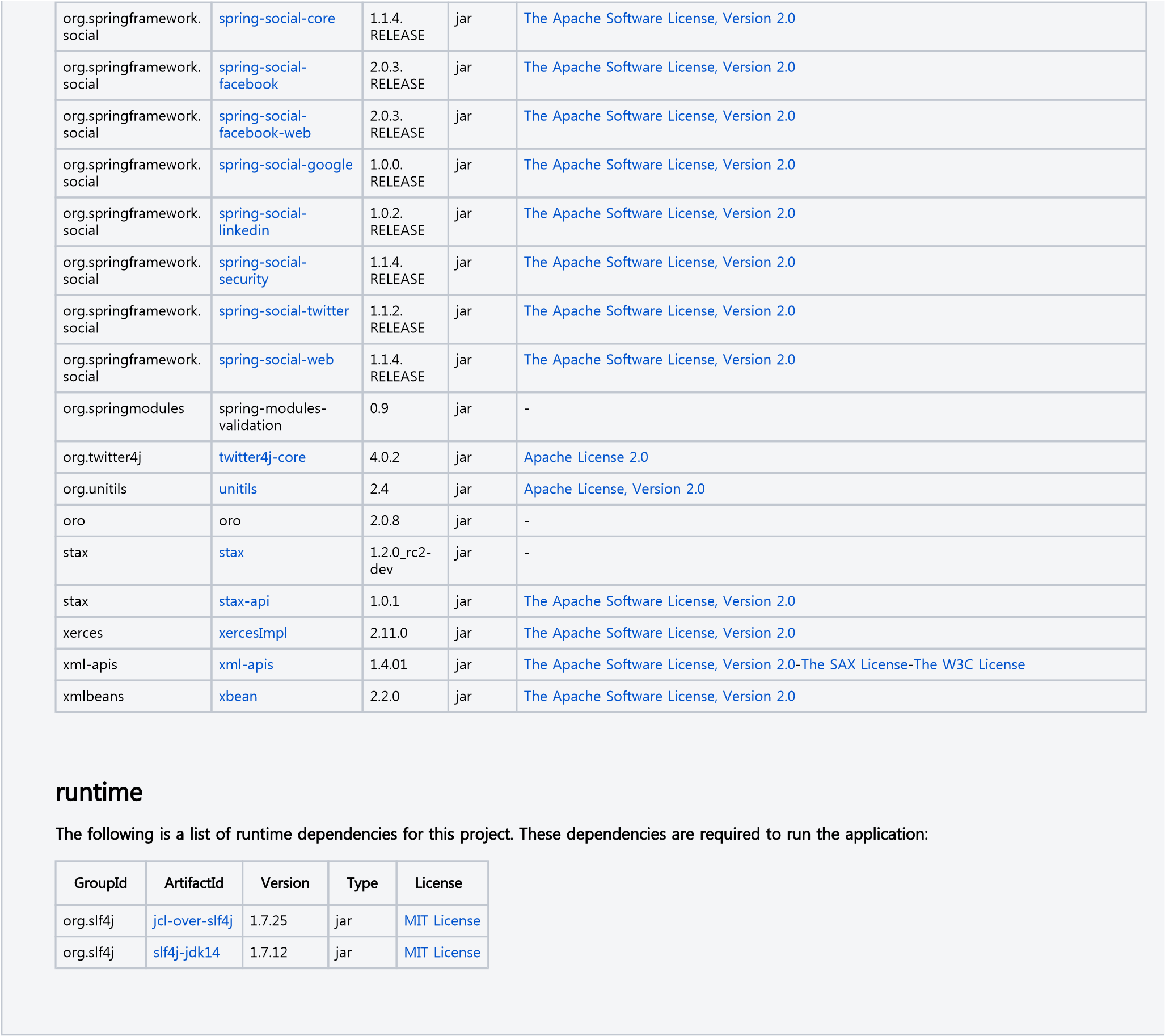
* [ASD3 - 오픈소스 라이선스 관리](https://docs.ahnlab.com/pages/viewpage.action?pageId=34571025) 문서를 참고하며, 준수한다.
* 본 프로젝트는 내부 시스템 프로젝트이므로, 오픈소스 라이선스 준수 대상에 포함되지 아니한다.
* 하지만 오픈소스 관리 목적으로 다음과 같이 템플릿 프로젝트의 라이선스를 관리한다.
* 하기 목록은 템플릿 프로젝트의 기능으로 자동으로 오픈소스 목록을 export 할 수 있다.











# 6 제품 기능

## 6.1 엔진 테스트

### 6.1.1 악성 샘플 진단 테스트

엔진별로 대응된 신규 / 기존 악성 샘플의 진단 여부를 확인한다.

-

신규 범위 : 배포 예정인 버전에 대한 대응 악성 샘플을 말한다.

-

기존 범위 : 이미 배포된 엔진에 대한 대응 악성 샘플을 말한다.

샘플 관리 : 샘플 생성 및 삭제 등의 관리는 빌드 스크립트가 주관하고 있으며 현재 18일치에 대한 샘플을 유지하고 있다.

(

샘플 관리 운영은 분석팀 에서 수행한다

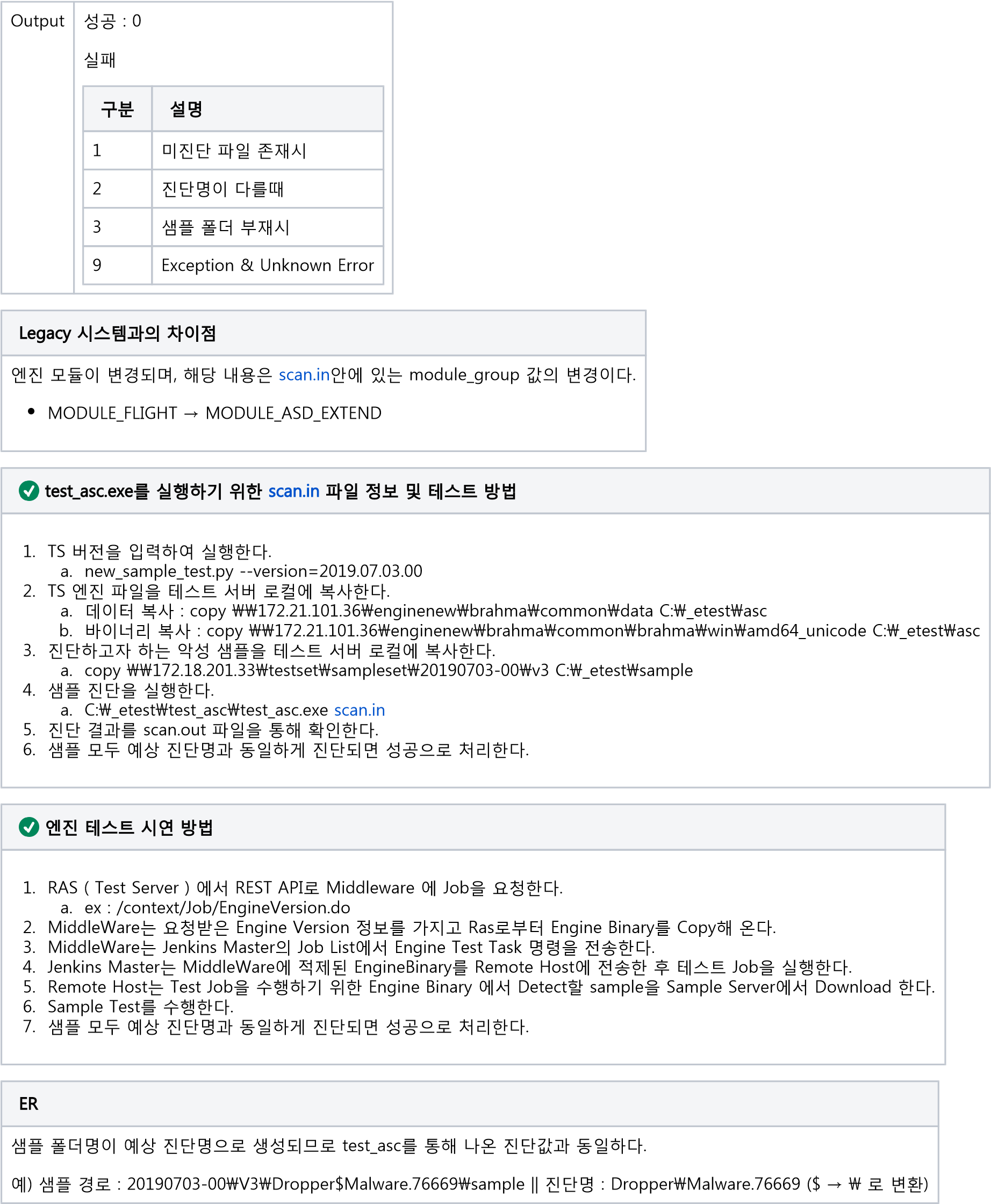
)

항목

값

Input

TS 엔진 버전 ( ex: 2019.07.03.00 )



6.1.1.1 엔진셋 복사 입력받은 TS 엔진 버전에 대한 엔진 파일을 테스트 서버 로컬에 복사한다.

엔진 파일은 oasis 서버로부터 가져온다.

패킹 과정에서 엔진 테스트용

셋을 제작

하여 전달 받는다.

테스트용 엔진 셋은 다음 컬럼의 이미지

에서 확인이 가능하다.

엔진셋 경로는

파일에서 engine\_path

[scan.i](http://scan.in/)

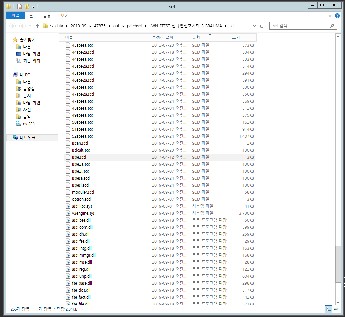
[n](http://scan.in/)

값으로 사용된다.

(

전달 받는 엔진 셋 예시

)



#### 6.1.1.2 샘플셋 확인

샘플서버는 SMB로 폴더 접근이 가능하며 접근시 사용되는 계정은 readonly 권한이 부여되어야 한다.

신

규 / 기존에 따라서 아래와 같이 확인해야 하는 샘플셋의 범위가 정해진다.

엔진의 테스트 대상 샘플의 개수는 한정하지 않는다. 단, 제공하는 샘플이 없는 경우는 Error 처리로 Exception처리 대상이다.

범위는 아래와 같이 정한다.

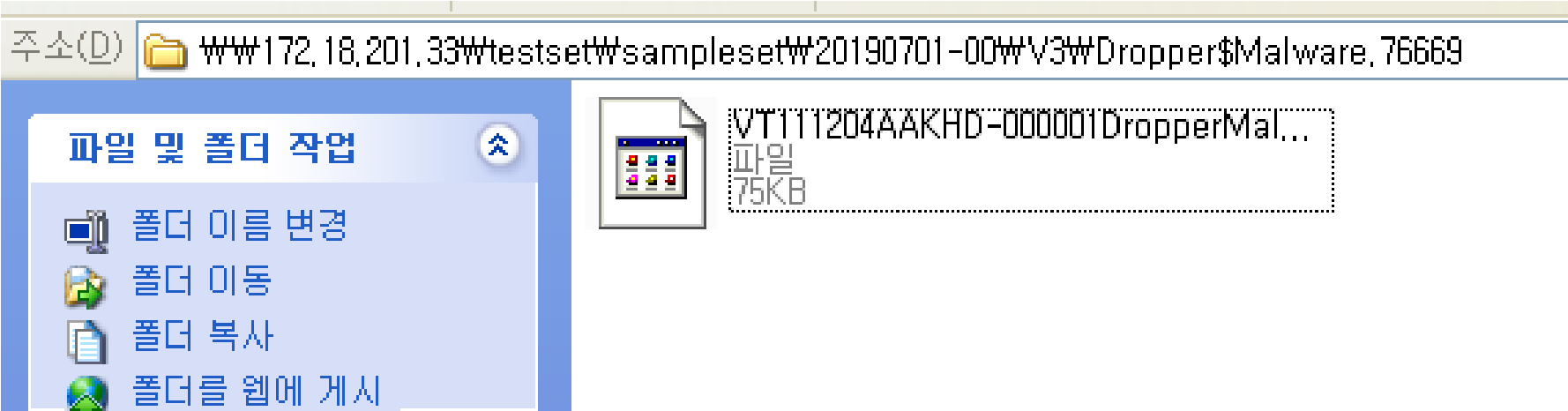
신규 : 샘플서버에서 입력받은 TS 엔진 버전으로 생성된 샘플 폴더 1개가 있는지 확인한다.

기존 : 샘플서버에서 입력받은 TS 엔진 버전보다 이전에 생성된 폴더가 최소 1개 이상 최대 10개가 있는지 확인한다.

폴더는 내림차순 기준으로 정렬된 범위이다.

샘플 BASE 위치 : \\172.18.201.33\testset\sampleset\

TS 버전별 샘플 폴더 경로 :



6.1.1.3 샘플셋 복사 샘플서버에서 TS 엔진 버전의 샘플 폴더를 테스트 서버의 로컬에 복사한다. 샘플셋 경로는 scan.in 파일에서 scan\_path 값으로 사용된다.

6.1.1.4 샘플셋 진단 아래의 scan.in 파일 내용으로 test\_asc.exe을 이용하여 악성 샘플을 진단한다.

test\_asc.exe를 실행하기 위한 설정 파일로 엔진 모듈, 엔진 경로, 스캔 경로 등과 같은 값들에 대한 설정을 할 수 있다.

scan.in

action\_start = 1

action\_count = 1

log\_level = 2

[1]

test\_unit = MultiScan

scan\_thread = 2

engine\_path = C:\\_etest\asc

module\_group= MODULE\_ASD\_EXTEND

module\_mode = MODULE\_MODE\_PRIME

OPTION\_BIT\_SCAN\_TYPE\_HARMFUL = 1

OPTION\_BIT\_FORCE\_DELETE\_TO\_CLEAN = 0

OPTION\_BIT\_SCAN\_TYPE\_MALICIOUS = 1

OPTION\_INT\_MESSAGE\_LOCALE = 2

input\_product\_string = v3lite

scan\_path = C:\\_etest\sample

6.1.2.1.1.2

Mupdate 은 상기와 같이 셋팅한다.

6.1.1.5

진단 결과 확인

scan.out 파일을 파싱하여 샘플별 진단 결과를 확인한다.

-

샘플 폴더명은 예상진단명으로 실제 test\_asc 실행시 나타나는 진단명과 동일한지 확인한다.

-

테스트에 사용된 샘플은 삭제한다.

(

테스트에 필요한 파일과 테스트를 진행하면서 생성된 산출물은 모두 middleware의 관리영역에 해당하며 테스트 종료시 일괄 삭

제한다 )

scan.out

\WINDOWS\\_sample\RTF$Downloader\EC190703AAAIL

-000001780362.

[2019/07/03 11:12:57][00] C:

docxx\_00000005\_00000000

->

[2168ad41] RTF/Downloader, [900] N/A, [00000a00] C\_ATTR\_DELETE\_FILE, C\_ATTR\_MODIFY\_REGISTRY

[2019/07/03 11:12:57][01] C:

\WINDOWS\\_sample\RTF$Exploit\EC190702AADXK

-

00000174153701fc7d03c0c6a03f6755230d5

f.docxx\_00000001\_00000000

->

[2164d602] RTF/Exploit, [900] N/A, [00000a00] C\_ATTR\_DELETE\_FILE, C\_ATTR\_MODIFY\_REGISTRY

[2019/07/03 11:12:57] -----------------------------------------

Thread Count Detect V-Count V-Detect

[2019/07/03 11:12:57]

[2019/07/03 11:12:57] 0 1 1 1

1

1

[2019/07/03 11:12:57] 1 1 1 1

[2019/07/03 11:12:57] -----------------------------------------

[2019/07/03 11:12:57] 2 2 2 2

6.1.1.6 결과 처리 테스트의 성공, 실패 여부를 확인한다.

성공 : 모든 파일이 예상 진단명으로 진단된다.

실패 : 하나 이상의 파일이 예상 진단명과 다르게 진단되거나 미진단이 있다.

미진단과 진단명 차이 발생시 Work를 자동으로 생성한다. ( Issue 생성은 Test 결과를 컨트롤하는 RAS에서 관리한다 ) 프로젝트명 : 엔진 대응 (ENGRES) 제목 : {엔진버전} 엔진 대응 지켜보는 사람 : 안랩 분석대응 분석팀 확인 후 진행될 수 있도록 상태를 CONFIRM 형태로 한다.

기존 샘플 진단 테스트에서 미진단 혹은 다른 진단명으로 진단되는 경우 분석팀에 확인 후 진행한다. 이 경우 성공 처리 후 다음 엔진으로 조치될 수 있다.

### 6.1.2 엔진 업데이트 테스트

제품 ( 현재는 V3 ES 9.0 : 이후 변경 가능성 있음 ) 업데이트기능을 이용하여 엔진 패치셋에 대한 무결성 검증 및 업데이트 기능을 확인한다.

- RAS 서비스 테스트를 위하여 할당받은 키를 사용하여 제품의 API를 활용하여 엔진 업데이트 테스트를 진행합니다.

항목

값

.

1

2

.

3

.

a.

b.

c.

d.

4

.

.

5

Input

API Key

API Key는 대외비이며, 유출시 V3개발팀으로 이슈를 공유해야 합니다.

대상은 V3 IS/ES/NET 90 제품의 컨트롤 key이며 기존 제품은 별도의 Key가 존재하지 않습니다.

키값은 다음을 통하여 획득 할 수 있습니다.

[https://svn.ahnlab.com/Service/IRService/Project/RAS/6.0/Trunk/Src/py/ras/util/sysmon\_control.p](https://svn.ahnlab.com/Service/IRService/Project/RAS/6.0/Trunk/Src/py/ras/util/sysmon_control.py)

[y](https://svn.ahnlab.com/Service/IRService/Project/RAS/6.0/Trunk/Src/py/ras/util/sysmon_control.py)

Output

명령( API 호출 )을 활용하여 엔진 버전을 체크 할 수 있습니다.

[

ENGINE\_VERSION

]

V3AV\_Engine=2013.09.04.02

IPS\_Engine=2012.09.26.00

-

Legacy와의 차이점

업데이트 모듈을 활용하는 테스트 방식에서 실제 제품의 API를 활용하여 업데이트 테스트와 무결성 검증을 동시에 진행한다.



업데이트 테스트 시연 방법

WAS Middleware 에 REST API 로 Update Test 이벤트를 트리깅 한다.

WAS Middleware 는 Jenkins Master 서버의 JOB 을 실행한다.

JOB에 등록된 다음의 실행을 수행한다. ( 다음은 Agent에서 수행하는 TASK List 이다 )

사용자 정의 업데이트 서버 설정을 Middleware 인 WAS 로 변경한다.

제품 업데이트 명령을 실행한다.

제품의 실시간 검사 상태값을 확인한다. ( 상태값 결과를 MiddleWare 에 기록한다 )

제품 엔진 버전값을 확인한다. ( 버전 결과를 MiddleWare에 기록한다 )

MiddleWare는 요청받은 엔진버전과 Agent의 명령 수행 결과를 비교하여 동일여부를

RAS 데이터베이스에 기록한다.

RAS 의 이벤트 트리깅에 의하여 데이터베이스 탐색 결과에 따라 엔진 빌드를

Continue 할지 STOP 할지를 결정한다.



ER : 제품의 기능을 활용하여 온전하게 업데이트가 수행 후 의도한 제품의 엔진 버전인지 확인한다.

#### 6.1.2.1 제품 업데이트 서버 설정

제품 API를 활용하여 업데이트 서버를 사용자 정의로 설정한다.

업데이트 서버 설정 API 실행 명령

# V3Exec.exe

# public case : "C:\Program Files\ahnlab\V3ES90\V3Exec.exe"

#

V3Exec.exe /key “1.1 API-KEY" /pU /u “http:// / ”

#### 6.1.2.2 제품 업데이트 실행

제품 API를 활용하여 업데이트 Event Trigging 을 명령한다.

업데이트 이벤트 API 실행 명령

# UI

V3Exec.exe /key “1.1 API-KEY" /eU

# Silent()

V3Exec.exe /key “1.1 API-KEY" /eU /bs

6.1.2.3 제품 실시간 검사 상태 확인 제품 API를 활용하여 실시간 검사 상태값을 확인한다.

( 제품 실시간 서비스 상태 확인 결과인 Output 파일은 INI format으로 생성된다 )

실시간 검사 상태값 확인 API 명령어

#

V3Exec.exe /key “1.1 API-KEY" /sC /output "\.txt"

실시간 검사 상태값 결과 ( Output )

[

USAGE

]

REALTIME\_MONITOR=1 #

USB\_DRIVE\_SCAN=0 # USB

V3\_SELF\_PROTECT=0 #

SYSTEM\_PROTECT=0 #

MDP\_MONITOR=0 #

REPU\_MONITOR=0 #

CLOUD\_MONITOR=0 #

WEB\_MONITOR=1 #

HIPS\_MONITOR=1 #

NIPS\_MONITOR=0 #

NETWORK\_FIREWALL=0 #

DEVICE\_CONTROL=0 #

6.1.2.4 제품 엔진 Version Check 제품 API를 활용하여 업데이트 된 엔진의 Version 을 확인한다.

( V3 ES 9.0 의 업데이트된 엔진 버전을 체크한다. Output 파일은 INI format으로 생성된다. )

제품에 적용된 엔진 버전을 확인하는 명령어

#

V3Exec.exe /key “1.1 API-KEY" /vC /output "\.txt"

엔진 버전 확인 결과 ( Output )

]

ENGINE\_VERSION

[

V3AV\_Engine=2013.09.04.02

IPS\_Engine=2012.09.26.00



업데이트 모듈을 활용한 테스트 ( legacy method )

6.1.2

엔진 업데이트 테스트

Mupdate와 Autoup등의 업데이트에 관련된 모듈을 이용하여 엔진 패치셋에 대한 무결성 검증을 확인한다.

-

이전 버전과 상관없이 해당 엔진 패치셋을 모두 다운로드 받는다.

-

델타파일의 업데이트는 지원하지 않는다.

항목

값

Input

Product Code

D3 : AhnLab V3 IS 90

Engine Verison

2019.07.23.00

Update Type

Mupdate

Autoup

Output

성공 : 0

실패

1

: 엔진 업데이트 실패

6

: 엔진 초기화 실패

9

:

Exception & Unknown Error

-

Legacy와의 차이점

1

.

a.

2

.

3

.

4

.

a.

5

.

a.

6

.

업데이트 모듈이 smartupdate → mupdate 로 변경되며, autoup 의 경우 UI → Silence 모드로 실행 타입이 변경

된다.

6.1.2.1

Mupdate 업데이트 테스트



업데이트 테스트 ILT 시연 방법

테스트 대상으로 하는 제품에 대한 제품코드값을 입력하여 실행한다.

update\_test.py --pdcode=d3 --engine-version=2019.07.23.00 --update-type=mupdate

엔진 업데이트 CDN 환경을 테스트 서버에 일회성으로 셋팅한다.

mupdate을 통해 엔진 업데이트를 실행한다.

다운로드 받은 파일과 원본 파일과의 무결성 체크를 sha256값과 파일 수정시간으로 비교한다.

원본파일의 정보는 fileinfo.txt 파일로 제공받는다.

다운로드 받은 엔진 파일로 제품에 해당하는 엔진 모듈 값을 통해 엔진 초기화를 진행한다.

제품별 엔진모듈 내용이 기록된 config.ini 파일을 함께 제공한다.

무결성 체크 및 엔진 초기화 과정에서 이슈가 없으면 성공으로 처리한다.



ER : 입력된 제품에 대한 엔진셋이 업데이트로 지정된 경로에 다운로드되며 해당 엔진셋으로 test\_asc을 통해 엔진

초기화 및 로딩이 되어야 한다.

6.1.2.1.1

환경 설정

6.1.2.1.1.1

가상의 CDN 환경을 구성한다.

-

구성 방법

테스트 서버 로컬에서 특정 경로에 엔진 패치셋을 복사한다.

복사된 경로가 http로 서비스 될 수 있도록 웹서버스를 제공한다. ( Web-WAS Middle ware를 활용한다.

)

엔진셋에는 최신 mupdate와 suarez 패치셋을 포함한다.

mupdate : patch\04

suarez : patch\05

6.1.2.1.1.2

Mupdate 은 아래와 같이 셋팅한다.

-

최신 mupdate 패치셋을 테스트 서버 로컬에 복사한다.

-

제품 코드별로 생성된 mupdate 설정 파일을 복사한다.

Mupdate.cfg

[

update

]

num\_update\_item=1 # . [UpdateItemXX] .

autoselect\_server=0

retry=15

down\_directory=C:\UpdateTest\Update\ #

country\_code=82

NewDownloadEngine=1

UpdateServerMode=1

Download\_oneexe\_mode=0

MainAppName=RAS\_ENGINE\_TEST

LogPath=C:\UpdateTest\ # MUpdate2

Use\_proxy=0 # 1 . 0 .

proxy\_auth\_id=id #

proxy\_auth\_pass=password #

Proxy\_ip=1.2.3.4 #

Proxy\_port=1234 #

TurnOffCRCCheck=0

FullCheck=1

CCCheck=1

UseStableEngine=0

Url=http://1.2.3.4/file/ #

[

UpdateItem00] #

Application=d3 # .

ProductCode=d3 #

Available=1 #

Win\_Product=1 # Windows 1, 0

6.1.2.1.1.2

Mupdate 을 실행한다.

1

.

.

2

a.

.

3

.

4

a.

b.

.

5

설정이 완료된 후에 mupdate2.exe 파일을 실행한다.

Mupdate 리턴코드 :

[w](https://docs.ahnlab.com/x/NeLHCw)

[https://docs.ahnlab.com/x/NeLHC](https://docs.ahnlab.com/x/NeLHCw)

리턴값이 0 이면 성공이고 엔진 파일 무결성 확인 단계로 넘어간다.

Autoup 업데이트 테스트

6.1.2.2



업데이트 테스트 ILT 시연 방법

silence 동작으로 실행되는 엔진 실행압축 파일을 생성 및 테스트 서버 로컬에 복사한다.

테스트 대상으로 하는 제품에 대한 제품코드값을 입력하여 실행한다.

update\_test.py --pdcode=d3 --engine-version=2019.07.23.00 --update-type=autoup

업데이트가 완료되면 cmd에서 echo %errorlevel% 명령을 통해 성공 여부를 확인한다. (리턴값 0 인 경우 성공)

Registry 안에서의 제품 정보에서 엔진 버전을 확인한다.

HKEY\_LOCAL\_MACHINE\SOFTWARE\AHNLAB\ASPack\9.0

AscVersion 키값 확인

엔진 버전이 동일하면 성공으로 처리한다.



ER : 엔진셋의 버전과 제품에서의 버전이 일치해야 한다.

6.1.2.2.1

환경 설정

6.1.2.2.1.1 V3

제품 설치 후 자동 업데이트 기능 OFF

6.1.2.2.1.2

Silence 모드의 실행압축파일을 생성한다.

NEA에서 command 로 실행시키지 위함으로 autoup 실행 옵션 중에 /s 가 적용된 실행압

축파일을 생성한다.

옵션 변경 : /ie → /as

6.1.3.2.1.3

Command 로 실행압축 파일을 실행한다.

실행 완료되면 echo %errorlevel% 을 실행하여 리턴 값을 확인한다.

Autoup 리턴코드 :

[https://docs.ahnlab.com/x/7xEH](https://docs.ahnlab.com/x/7xEHD)

[D](https://docs.ahnlab.com/x/7xEHD)

리턴값이 0 이면 성공이고 제품에서의 엔진 버전을 확인한다.

6.1.3.2.3

제품의 엔진버전을 확인한다.

제품 Registry에서 AscVersion을 통해 엔진버전을 확인한다.

String value = WinRegistry.readString (

WinRegistry.HKEY\_LOCAL\_MACHINE,

//HKEY

"SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion",

//Key

"ProductName");

//ValueName

System.out.println("Windows Distribution = " + value);

입력한 엔진버전과 동일한지 확인한다.

6.1.2.3

엔진 파일 무결성 확인

업데이트 모듈에 의해 다운로드된 엔진 파일과 실제 원본의 엔진 파일을 아래의 키값으로 비교한다.

-

sha256

-

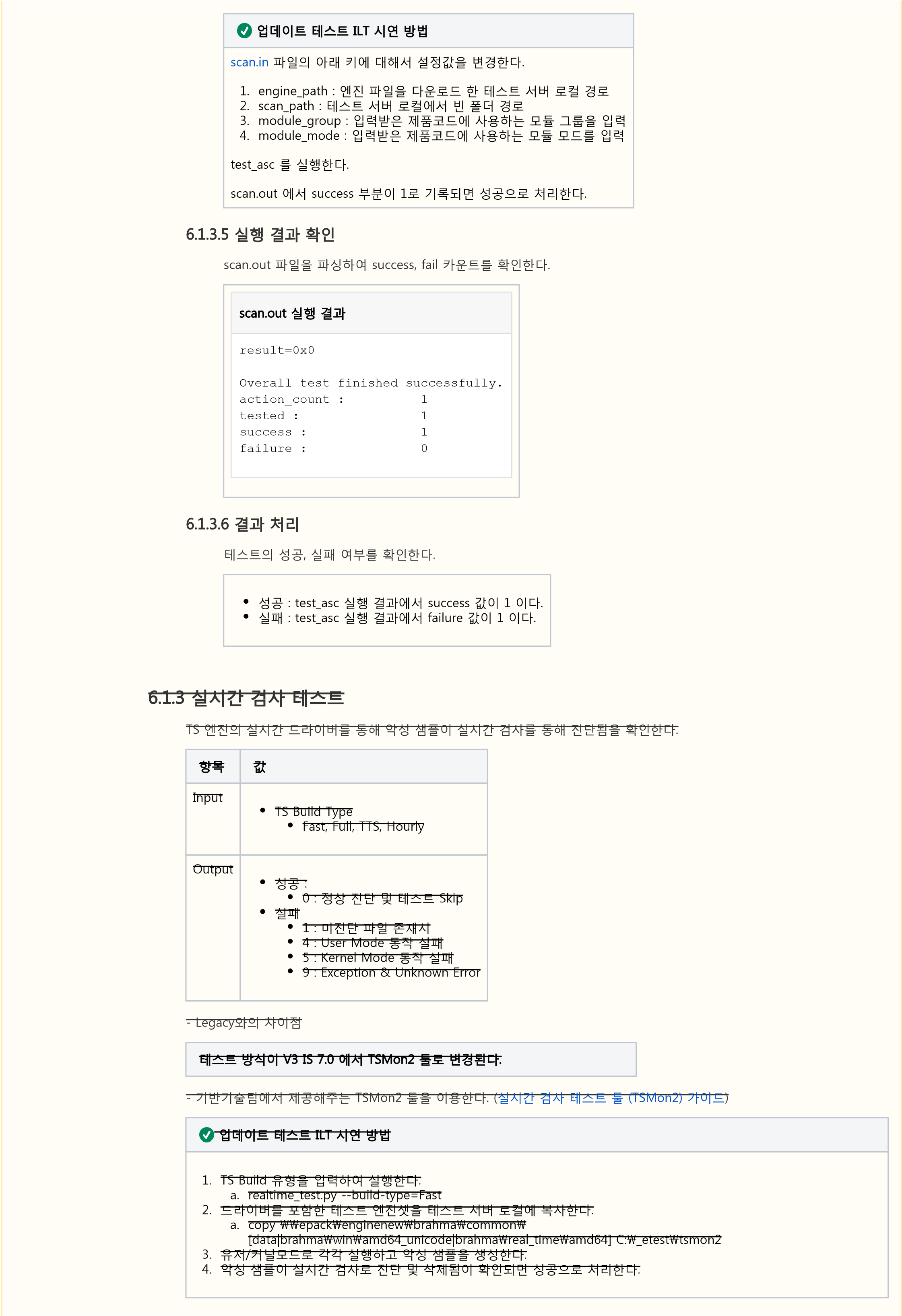
modify date

6.1.3.4

엔진 초기화

다운로드 받은 엔진으로 제품에서 사용하는 엔진 모듈값으로 초기화가 되는지 확인한다.

test\_asc를 사용하여 확인하며 순서는 아래와 같다.



6.1.3

엔진 모듈 초기화

TBD

엔진 모듈에 대해서 제품별 정상적으로 동작하는지 확인한다.

.

1

2

.

.

3

4

.

5

.

.

6

.

1

2

.

.

3

4

.

5

.

.

6



~~ER : 악성 샘플 생성시 실시간 검사가 정상적으로 동작하여 샘플이 삭제된다.~~

~~빌드 유형 체크~~

~~6.1.2.1~~

~~테스트 진행 여부를 결정하기 위해 입력받은 TS Build Type을 확인한다.~~

~~-~~

~~Fast, Full : 테스트 진행~~

~~Hourly, TTS : 테스트 Skip~~

~~-~~

~~6.1.2.2~~

~~엔진셋 복사~~

~~실시간 검사 드라이버를 포함한 엔진셋을 테스트 서버 로컬에 복사한다.~~

~~에서 '3.1 관련 모듈' 을 참고한다.~~

[~~드~~](https://docs.ahnlab.com/pages/viewpage.action?pageId=65638227)

[~~실시간 검사 테스트 툴 (TSMon2) 가이~~](https://docs.ahnlab.com/pages/viewpage.action?pageId=65638227)

~~엔진 파일은 epack 서버로부터 가져오며 바이너리 아키텍처는 x86, x64로 구분한다. (arm64는 추후 적용을 고려한다.~~

)

NEXT

~~scd : enginenew\braham\brahma\common\data~~

~~-~~

~~dll : enginenew\brahma\brahma\common\brahma\win\{amd64\_unicode|win32\_unicode}~~

~~-~~

~~realtime : D:\enginenew\brahma\brahma\common\brahma\real\_time\{amd64|win2k}~~

~~-~~

~~유저(User) 모드 실행~~

~~6.1.2.3~~

~~유저 모드로 실시간 검사를 실행하여 악성 샘플이 진단되는지 확인한다.~~

~~아래는 소프트웨어 QA팀으로 전달받은 시연 방법으로 bat 파일 또한 기반기술팀으로 제공받은 파일이다.~~



~~업데이트 테스트 ILT 시연 방법~~

~~TC\_01\_install.bat 파일을 실행한다.~~

~~TC\_02\_start\_user.bat 파일을 실행한다.~~

~~TC\_05\_getversion.bat 파일을 실행하여 엔진 버전을 확인한다.~~

~~악성 샘플 접근을 시도한다.~~

~~TC\_03\_stop.bat 파일을 실행한다.~~

~~TC\_04\_uninstall.bat 파일을 실행하여 종료한다.~~

~~커널(Kenel) 모드 실행~~

~~6.1.2.4~~

~~커널 모드로 실시간 검사를 실행하여 악성 샘플이 진단되는지 확인한다.~~

~~아래는 소프트웨어 QA팀으로 전달받은 시연 방법으로 bat 파일 또한 기반기술팀으로 제공받은 파일이다.~~



~~업데이트 테스트 ILT 시연 방법~~

~~TC\_01\_install.bat 파일을 실행한다.~~

~~TC\_02\_start.bat 파일을 실행한다.~~

~~TC\_05\_getversion.bat 파일을 실행하여 엔진 버전을 확인한다.~~

~~악성 샘플 접근을 시도한다.~~

~~TC\_03\_stop.bat 파일을 실행한다.~~

~~TC\_04\_uninstall.bat 파일을 실행하여 종료한다.~~

~~실행 결과 확인~~

~~6.1.2.5~~

~~에서 '3.4.1 디버그 로그 파일' 을 참고한다.~~

[~~실시간 검사 테스트 툴 (TSMon2) 가이~~](https://docs.ahnlab.com/pages/viewpage.action?pageId=65638227)

[~~드~~](https://docs.ahnlab.com/pages/viewpage.action?pageId=65638227)

~~scanlog.log 파일을 파싱하여 악성 샘플에 대한 진단 여부를 확인한다.~~

~~6.1.2.6~~

~~결과 처리~~

~~테스트의 성공, 실패 여부를 확인한다.~~

~~-~~

~~성공 : 악성 샘플이 정상적으로 진단된~~

다.

~~실패 : 악성 샘플이 미진단 된다.~~

~~-~~

6.1.3.1 제품별로 설정된 엔진모듈 현황을 정리한다.

6.1.3.2 ahn.ui 파일에서 제품별 파일 그룹을 정리한다.

6.1.3.3 엔진 패치셋에서 제품별 엔진 파일을 취합하여 구성한다.

6.1.3.4 test\_asc를 통해 엔진 초기화 및 로딩을 해본다.

## 6.2 테스트 인프라

### 6.2.1 테스트 환경 설정

#### 6.2.1.1 개발 및 TEST 목적의 RAS Server 구축 및 활용 6.2.1.2 개발 및 TEST 목적의 Oasis Storage 구축 및 활용 ( epack & NEA ) 6.2.1.3 개발 및 TEST 목적의 WEB - WAS Server 구축 및 활용 6.2.1.4 개발 및 TEST 목적의 Jenkins Master - Agent 구축 및 활용

* 인프라 정보 : [1. RAS - 시스템 정보](https://docs.ahnlab.com/pages/viewpage.action?pageId=125469564)
* 시스템 구축문서 : [2.2. EngineTestSystem Infra](https://docs.ahnlab.com/display/ASCM/2.2.+EngineTestSystem+Infra)

## 6.3 가상화 환경 컨트롤

### 6.3.1 가상 이미지 관리

#### 6.3.1.1 개발기간 가상 이미지 관리 6.3.1.2 가상 이미지 운영 전환 전략 6.3.1.3 개발 가상이미지 구성 Set 과 운영 가상이미지 구성 Set 관리 방안

- [3.3. ETS - 시스템 운영 정책](https://docs.ahnlab.com/pages/viewpage.action?pageId=234897940) 하위에 항목을 만들어서 관리하도록 하겠습니다.

~~6.4 엔진 업데이트 환경 설정~~

~~6.4.1 가상 업데이트 서버 관리 6.4.2 업데이트 모듈 설정~~

## 6.5 모니터링

### 6.5.1 테스트 서버 상태 확인

* [http://172.21.10.58](http://172.21.10.58/) grafana &influxdb 인프라를통해서 시스템 매트릭 정보를 수집 하거나 알림 받을 수 있다 ( 현재는 E-Mail을 통해 시스템이 이슈를 알림 받을 수 있다.)
* 서비스 매트릭은 모니터링 고도화 작업시 진행을 한다.

TBD

( [Notification Service](https://docs.ahnlab.com/display/ASDI/Notification+Service) 사용을 검토하여 중요한 알림을 SMS, Teams를 통해서도 수신할수 있도록 한다)

### 6.5.2 테스트 진행 상태 확인

RAS를 통해 확인하며 상태는 아래와 같다.

구분 설명

FINISH 성공

ERROR 실패

CONFIRM 확인 필요

READY 대기중

WORKING 진행중

## 6.6. 기타

### 6.6.1. V3 제품 컨트롤

- [RAS-V3Ctl.md](https://docs.ahnlab.com/download/attachments/219029485/RAS-V3Ctl.md?version=1&modificationDate=1569310239000&api=v2)

6.6.2. RAS 시스템 전체 구성도 ( ETS 표시됨 . publish by [신원두/ASD서비스개발팀](https://docs.ahnlab.com/display/~wondu) )

# 1. 대응 시스템 구성도

