

인터페이스 구현

시스템 인터페이스 요구사항 분석

- ❖ 시스템 인터페이스: 독립적으로 떨어져 있는 시스템끼리 서로 연동하여 상호 작용하기 위한 접속 방법이나 규칙
- ❖ 시스템 인터페이스 요구사항 명세서의 구성 요소
 - ✓ 인터페이스 이름
 - ✓ 연계 대상 시스템
 - ✓ 연계 범위 및 내용
 - ✓ 연계 방식
 - ✓ 송신 데이터
 - ✓ 인터페이스 주기
 - ✓ 기타 고려사항
- ❖ 시스템 인터페이스 요구사항 분석
 - ✓ 시스템 인터페이스 요구사항 분석은 요구사항을 분류하고 구체적으로 명세 한 후 이해 관계자에게 전달하는 일련의 과정
 - ✓ 소프트웨어 요구사항 분석 기법을 적절히 이용
 - ✓ 요구사항의 분해가 필요한 경우 적절한 수준으로 세분화
 - ✓ 요구사항 분석 시 누락한 요구사항이나 제한 조건을 추가
 - ✓ 요구사항에 대한 상대적 중요도를 평가하여 우선순위를 부여

시스템 인터페이스 요구사항 분석

❖ 시스템 인터페이스 요구사항 분석 절차

- ✓ 요구사항을 선별하여 별도로 요구사항 목록을 생성
- ✓ 요구사항과 관련된 자료를 준비
- ✓ 기능적인 요구사항과 비기능적인 요구사항을 분류
- ✓ 요구사항을 분석하고 요구사항 명세서에 내용을 추가하거나 수정
- ✓ 추가 수정한 요구사항 명세서와 요구사항 목록을 관련 이해관계자에게 전달



인터페이스 요구사항 검증

❖ 요구 사항 검증

- ✓ 인터페이스의 설계 및 구현 전에 사용자들의 요구사항이 요구사항 명세서에 정확하고 완전하게 기술되었는지 검토하고 개발 범위의 기준인 베이스라인을 설정하는 것
- ✓ 인터페이스의 설계 및 구현 중에 요구사항 명세서의 오류가 발견되어 이를 수정할 경우 많은 비용이 소요되므로 프로젝트에서 매우 중요
- ✓ 검증 수행 순서
 - ❑ 요구사항 검토 계획 수립
 - ❑ 검토 및 오류 수정
 - ❑ 베이스라인 설정

❖ 요구 사항 검증 방법

- 요구사항 검토: 요구사항 명세서의 오류 확인 및 표준 준수 여부 등의 결함 여부를 검토 담당자들이 수작업으로 분석하는 방법으로 동료 검토(Peer Review), 워크 스루(Walk Through), Inspection 등이 있음
- 프로토타이핑 (Prototyping): 사용자의 요구사항을 정확히 파악하기 위해 실제 개발될 소프트웨어에 대한 견본품 (Prototype)을 만들어 최종 결과물을 예측
- 테스트 설계: 요구사항은 테스트할 수 있도록 작성되어야 하며 이를 위해 테스트 케이스 (Test Case)를 생성하여 이후에 요구사항이 현실적으로 테스트 가능한지를 검토
- CASE 도구 활용: 일관성 분석(Consistency Analysis)을 통해 요구사항 변경사항의 추적 및 분석, 관리, 표준 준수 여부를 확인

인터페이스 요구사항 검증

❖ 시스템 인터페이스 요구사항 명세서

✓ 요구 사항 검토

- 동료 검토(Peer Review): 요구사항 명세서 작성자가 명세서 내용을 직접 설명하고 동료들이 이를 들으면서 결함을 발견하는 형태의 검토 방법
- Walk Through: 검토 회의 전에 요구사항 명세서를 미리 배포하여 사전 검토한 후에 짧은 검토 회의를 통해 결함을 발견하는 형태의 검토 방법
- Inspection: 요구사항 명세서 작성자를 제외한 다른 검토 전문가들이 요구사항 명세서를 확인하면서 결함을 발견하는 형태의 검토 방법

✓ 요구 사항 검토 항목

- 완전성(Completeness): 사용자의 모든 요구사항이 누락되지 않고 완전하게 반영
- 일관성(Consistency): 요구사항이 모순되거나 충돌되는 점 없이 일관성을 유지
- 명확성(Unambiguity): 모든 참여자가 요구사항을 명확히 이해
- 기능성(Functionality): 요구사항이 '어떻게(How to)' 보다 '무엇을(What)'에 중점
- 검증 가능성(Verifiability): 요구사항이 사용자의 요구를 모두 만족하고, 개발된 소프트웨어가 사용자의 요구 내용과 일치하는지
- 추적 가능성(Traceability): 요구사항 명세서와 설계서를 추적
- 변경 용이성(Easily Changeable): 요구사항 명세서의 변경이 쉽도록

인터페이스 시스템 식별

- ❖ 인터페이스 시스템 식별은 인터페이스별로 인터페이스에 참여하는 시스템들을 송신 시스템과 수신 시스템으로 구분하여 작성하는 것
- ❖ 인터페이스 시스템 식별을 위한 선행 작업
 - ✓ 개발 시스템과 내 • 외부 시스템 식별
 - ✓ 내 • 외부 시스템 환경 및 관리 주체 식별
 - 내 • 외부 시스템의 실제 운용 환경과 하드웨어 관리 주체를 확인
 - 내 • 외부 시스템 환경: 연계할 시스템 접속에 필요한 IP 또는 URL, Port 정보 등 시스템의 실제 운용 환경
 - 내 • 외부 시스템 관리 주체: 하드웨어 관리 담당자
 - ✓ 내 • 외부 시스템 네트워크 연결 정보 식별
 - ✓ 인터페이스 식별



송 · 수신 데이터 식별

❖ 식별 대상 데이터

- ✓ 송 · 수신 시스템 사이에서 교환되는 데이터로 규격화된 표준 형식에 따라 전송되는 데이터
- ✓ 교환되는 데이터 종류
 - 인터페이스 표준 항목
 - 송 · 수신 데이터 항목: 업무를 수행하는 데 사용하는 데이터
 - 공통 코드: 시스템에서 공통으로 사용하는 코드

❖ 인터페이스 표준 항목

- ✓ 송 · 수신 시스템을 연계하는데 표준적으로 필요한 데이터
- ✓ 시스템 공통부: 시스템 간 연동 시 필요한 공통 정보
- ✓ 거래 공통부: 시스템들이 연동된 후 송 · 수신되는 데이터를 처리할 때 필요한 정보



인터페이스 방법 명세화

❖ 인터페이스 방법 명세화

- ✓ 인터페이스 명세화 내용
 - 송 • 수신 방법 명세화
 - 송 • 수신 데이터 명세화
 - 오류 식별 및 처리 방안 명세화
- ✓ 필요 정보
 - 시스템 연계 기술
 - 인터페이스 통신 유형
 - 처리 유형
 - 발생 주기

❖ 시스템 연계 기술

- ✓ DB Link: 데이터베이스에서 제공하는 Link를 이용하는 방식
- ✓ API/Open API: 송신 시스템의 데이터베이스에서 데이터를 읽어와 제공하는 애플리케이션 프로그래밍 인터페이스 프로그램
- ✓ 연계 솔루션: EAI 서버와 송 • 수신 시스템에 설치되는 클라이언트를 이용하는 방식
- ✓ Socket: 소켓을 생성하여 포트를 할당하고 클라이언트의 통신 요청 시 클라이언트와 연결하여 통신하는 네트워크 기술
- ✓ Web Service: WSDL, UDDI, SOAP 프로토콜을 이용하여 연계하는 서비스

인터페이스 방법 명세화

- ❖ 인터페이스 통신 유형
 - ✓ 단방향
 - ✓ 동기
 - ✓ 비동기
- ❖ 처리 유형
 - ✓ 실시간 방식
 - ✓ 지연 처리 방식
 - ✓ 배치 방식



미들웨어 솔루션

❖ 미들웨어

- ✓ 운영체제와 응용 프로그램 또는 서버와 클라이언트 사이에서 다양한 서비스를 제공하는 소프트웨어
- ✓ 표준화된 인터페이스를 제공함으로써 시스템 간의 데이터 교환에 일관성을 보장
- ✓ 종류
 - ❑ DB
 - ❑ RPC(Remote Procedure Call): 원격 프로시저를 로컬 프로시저처럼 호출하는 미들웨어
 - ❑ MOM(Message Oriented Middleware): 비동기형 메시지를 전달하는 미들웨어
 - ❑ TP-Monitor(Transaction Processing Monitor): 트랜잭션 처리 및 감시하는 미들웨어
 - ❑ ORB(Object Request Broker): CORBA 표준 스펙을 구현한 객체 지향 미들웨어
 - ❑ WAS(Web Application Server): 웹 서버에서 동적인 콘텐츠를 제공하기 위한 미들웨어



모듈간 공통 기능 및 데이터 인터페이스

- ❖ 모듈 간 공통 기능 및 데이터 인터페이스 확인 절차
 - ✓ 인터페이스 설계서를 통해 모듈별 기능을 확인
 - ✓ 외부 및 내부 모듈을 기반으로 공통으로 제공되는 기능과 각 데이터의 인터페이스를 확인
- ❖ 인터페이스 설계서
 - ✓ 이 기종 시스템 또는 컴포넌트 간 데이터 교환 및 처리를 위한 목적으로 인터페이스 목록에 있는 각 시스템의 교환 및 데이터 및 업무, 송수신 주체 등을 정의한 문서
 - ✓ 종류
 - 일반적인 인터페이스 설계서
 - 시스템 인터페이스 설계서: 시스템의 인터페이스 현황을 확인하기 위하여 시스템이 갖는 인터페이스 목록과 인터페이스 명세를 보여주는 설계 문서
 - 상세 기능별 인터페이스 명세서: 인터페이스를 통한 각 세부 기능의 개요, 세부 기능이 동작하기 전에 필요한 사전 조건, 사후 조건 및 인터페이스 파라미터(데이터), 호출 이후 결과를 확인하기 위한 반환 값 등을 정의한 문서
 - 정적·동적 모형을 통한 인터페이스 설계서
 - 정적, 동적 모형으로 각 시스템의 구성 요소를 표현한 다이어그램을 이용하여 만든 문서
 - 시스템을 구성하는 주요 구성 요소 간 트랜잭션을 보여주고 해당 인터페이스가 시스템에서 어디에 속하고 해당 인터페이스를 통해 상호 교환되는 트랜잭션의 종류를 확인할 수 있다

모듈간 공통 기능 및 데이터 인터페이스

- ❖ 모듈 간 공통 기능 및 데이터 인터페이스 확인 절차
 - ✓ 인터페이스 설계서를 통해 모듈별 기능을 확인
 - ✓ 외부 및 내부 모듈을 기반으로 공통으로 제공되는 기능과 각 데이터의 인터페이스를 확인
- ❖ 인터페이스 설계서
 - ✓ 이 기종 시스템 또는 컴포넌트 간 데이터 교환 및 처리를 위한 목적으로 인터페이스 목록에 있는 각 시스템의 교환 및 데이터 및 업무, 송수신 주체 등을 정의한 문서
 - ✓ 종류
 - 일반적인 인터페이스 설계서
 - 시스템 인터페이스 설계서: 시스템의 인터페이스 현황을 확인하기 위하여 시스템이 갖는 인터페이스 목록과 인터페이스 명세를 보여주는 설계 문서
 - 상세 기능별 인터페이스 명세서: 인터페이스를 통한 각 세부 기능의 개요, 세부 기능이 동작하기 전에 필요한 사전 조건, 사후 조건 및 인터페이스 파라미터(데이터), 호출 이후 결과를 확인하기 위한 반환 값 등을 정의한 문서
 - 정적·동적 모형을 통한 인터페이스 설계서
 - 정적, 동적 모형으로 각 시스템의 구성 요소를 표현한 다이어그램을 이용하여 만든 문서
 - 시스템을 구성하는 주요 구성 요소 간 트랜잭션을 보여주고 해당 인터페이스가 시스템에서 어디에 속하고 해당 인터페이스를 통해 상호 교환되는 트랜잭션의 종류를 확인할 수 있다

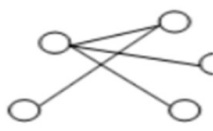

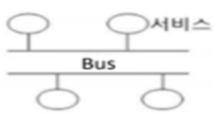
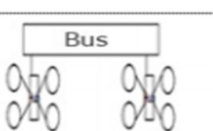
인터페이스 기능 식별

❖ 외부 및 내부 모듈 연계를 위한 인터페이스 기능 식별

✓ 내부, 외부 모듈 연계 방법(EAI, ESB, Web Service)

● EAI(Enterprise Application Integration)

- 기업에서 운영되는 플랫폼 및 애플리케이션들 간의 정보 전달, 연계, 통합을 가능하게 해주는 솔루션으로 EAI를 사용함으로써 각 비즈니스 간 통합 및 연계성을 증대시켜 효율성을 높여 줄 수 있으며 각 시스템 간의 확장성을 높여 줄 수 있음
- EAI는 크게 Peer to Peer형, Hub & Spoke 방식, Bus 형태의 구성으로 분류

유형	개념도	설명	특성
Point-to-Point		- 중간에 미들웨어를 두지 않고 각 애플리케이션 간 Point to Point 형태로 연결	- 솔루션 구매 없이 통합 - 상대적 저렴하게 통합 가능 - 변경, 재사용 어려움
Hub & Spoke		- 단일 접점이 허브 시스템을 통해 데이터를 전송하는 중앙 집중적 방식	- 모든 데이터 전송 보장 - 확장, 유지 보수 용이 - 허브 장애 시 전체 영향
Message Bus (ESB 방식)		- 애플리케이션 사이 미들웨어(버스)를 두어 처리 - 미들웨어 통한 통합	- 어댑터가 각 시스템과 버스를 두어 연결하므로 뛰어난 확장성, 대용량 처리 가능
Hybrid		- 그룹 내에는 Hub & Spoke 방식을 그룹 간 메시징 버스 방식을 사용	- 표준 통합 기술, 데이터 병목 현상 최소화

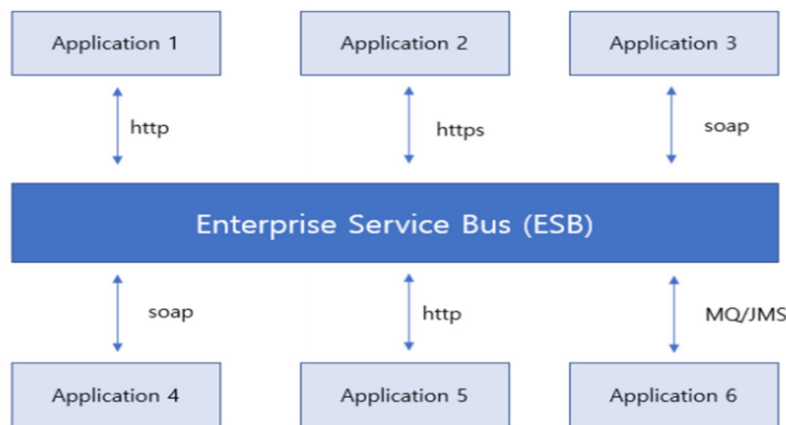
인터페이스 기능 식별

❖ 외부 및 내부 모듈 연계를 위한 인터페이스 기능 식별

✓ 내부, 외부 모듈 연계 방법

● ESB(Enterprise Service Bus)

- 애플리케이션 간 연계, 데이터 변환, 웹 서비스 지원 등 표준 기반의 인터페이스를 제공하는 솔루션
- 웹 서비스 중심으로 표준화된 데이터, 버스를 통해 이 기종 애플리케이션을 유연 (loosely-coupled)하게 통합하는 핵심 플랫폼(기술)
- ESB는 Bus를 중심으로 각각 프로토콜이 호환이 되게끔 변환이 가능하고 서비스 중심으로 메시지 이동을 라우팅 할 수 있다.
- 관리 및 보안이 쉽고 높은 수준의 품질 지원이 가능하지만 표준화가 미비하고 특정 벤더 종속, 성능 문제에서 개선되어야 할 부분이 남아 있다.



인터페이스 기능 식별

❖ 외부 및 내부 모듈 연계를 위한 인터페이스 기능 식별

✓ 내부, 외부 모듈 연계 방법

● Web Service

- SOAP: HTTP, HTTPS, SMTP 등을 활용하여 XML 기반의 메시지를 네트워크 상에서 교환하는 프로토콜
- WSDL: 웹 서비스명, 서비스 제공 위치, 프로토콜 등 웹 서비스에 대한 상세 정보를 XML 형식으로 구현
- UDDI: WSDL을 등록하여 서비스와 서비스 제공자를 검색하고 접근하는데 사용



인터페이스 데이터 표준 확인

- ❖ 모듈 간 인터페이스에 사용되는 데이터의 형식을 표준화하는 것
- ❖ 표준 확인 절차
 - ✓ 데이터 인터페이스 확인
 - ✓ 인터페이스 기능 확인
 - ✓ 인터페이스 데이터 표준 확인



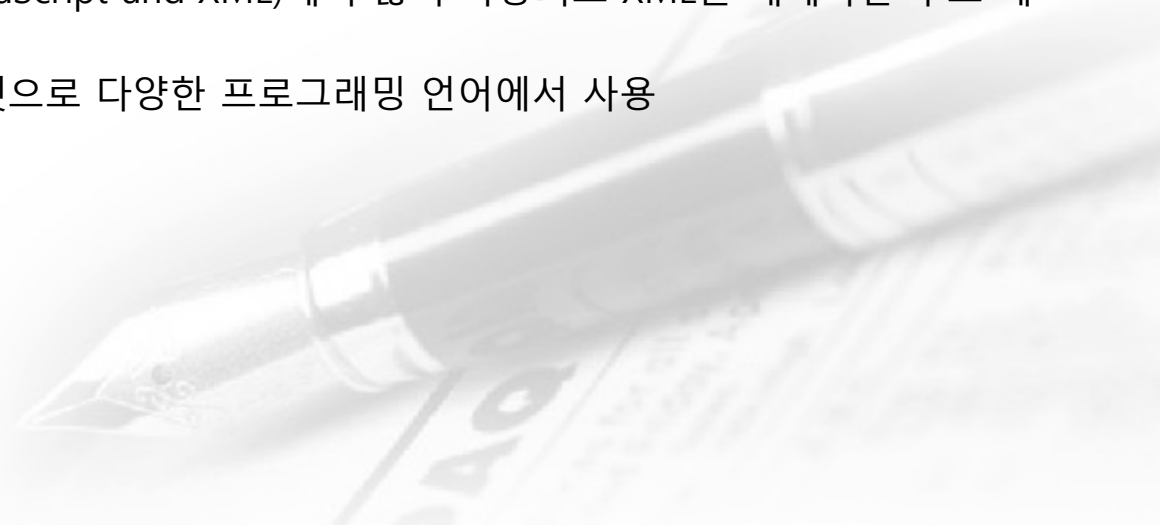
인터페이스 기능 구현 정의

- ❖ 인터페이스를 실제로 구현하기 위해 인터페이스 기능에 대한 구현 방법을 기능별로 기술하는 것
- ❖ 인터페이스 기능 구현 정의 절차
 - ✓ 컴포넌트 명세서 확인
 - ✓ 인터페이스 명세서 확인
 - ✓ 일관된 인터페이스 기능 구현 정의
 - ✓ 정의된 인터페이스 기능 구현 정형화
- ❖ 모듈 세부 설계서
 - ✓ 컴포넌트 명세서
 - ✓ 인터페이스 명세서



인터페이스 구현

- ❖ 송 • 수신 시스템 간의 데이터 교환 및 처리를 실현해 주는 작업
- ❖ 데이터 포맷
 - ✓ XML(eXtensible Markup Language)
 - ❑ 다른 특수한 목적을 갖는 마크업 언어를 만드는 데 사용하도록 권장하는 다목적 마크업 언어로 SGML의 간략화 된 버전으로 다른 많은 종류의 데이터를 기술하는 데 사용될 수 있으며, 다른 종류의 시스템끼리 데이터를 쉽게 주고받을 수 있게 하는 포맷
 - ❑ HTML, SGML(멀티미디어 전자문서들을 이 기종 시스템에 전송하기 위한 기술), VRML 등이 유사한 형태
 - ✓ JSON(JavaScript Object Notation)
 - ❑ JSON은 속성-값 쌍(attribute-value pairs)으로 이루어진 데이터 오브젝트를 전달하기 위해 사용하는 개방형 표준 포맷
 - ❑ AJAX(Asynchronous JavaScript and XML)에서 많이 사용되고 XML을 대체하는 주요 데이터 포맷
 - ❑ 언어 독립형 데이터 포맷으로 다양한 프로그래밍 언어에서 사용



인터페이스 구현

❖ 인터페이스 구현 절차

- ✓ 데이터를 각 시스템 환경에 맞게 선택
- ✓ 데이터 포맷을 이용해 인터페이스 객체를 생성
- ✓ 수신 측으로 전송
- ✓ 클라이언트에서 AJAX 등의 기술을 이용해서 수신 한 후 파싱 한 후 처리
- ✓ 수신 측에서 처리 결과를 송신 측으로 전송

❖ 인터페이스 처리 유형

- ✓ 실시간 방식: 사용자가 요청한 내용을 바로 처리해야 할 때 사용하는 방식
- ✓ 지연 처리 방식: 데이터를 매 건 단위로 처리할 경우 비용이 많이 발생할 때 사용하는 방식으로 나중에 처리하는 방식
- ✓ 배치 방식: 대량의 데이터를 처리할 때 모아서 처리하는 방식
- ✓ AJAX
 - ❑ 브라우저가 가지고있는 XMLHttpRequest 객체를 이용해서 전체 페이지를 새로 고치지 않고도 페이지의 일부만을 위한 데이터를 로드하는 기술
 - ❑ JavaScript를 사용한 비동기 통신, 클라이언트와 서버간에 XML 데이터를 주고받는 기술

인터페이스 예외 처리

❖ 인터페이스 예외 처리

✓ 데이터 통신을 이용한 방법

❑ 인터페이스 객체의 송·수신 시 발생할 수 있는 예외 케이스를 정의하고 각 예외 케이스마다 예외 처리 방법을 기술

❑ 인터페이스 동작이 실패할 경우를 대비한 것

✓ 인터페이스 엔티티를 이용한 인터페이스 예외 처리: 엔티티에 인터페이스의 실패 상황과 원인 등을 기록하고 이에 대한 조치를 취할 수 있도록 사용자 및 관리자에게 알려주는 방식으로 예외 처리 방법을 정의



인터페이스 보안

❖ 인터페이스 구현

✓ 인터페이스 보안 기능 적용

□ 네트워크 영역

- 인터페이스는 시스템 모듈 간 통신 및 정보 교환을 지원하므로 데이터 변조 · 탈취 및 인터페이스 모듈 자체의 보안 취약점이 있을 수 있음
- 스니핑(Sniffing)을 이용한 데이터 탈취 및 위협을 방지하기 위해 네트워크 트래픽에 대한 암호화를 설정
- 암호화는 IPSec, SSL, S-HTTP 등의 다양한 방식으로 적용



인터페이스 보안

❖ 인터페이스 구현

✓ 인터페이스 보안 기능 적용

□ 애플리케이션 영역

- 시큐어 코딩 대표적인 웹 애플리케이션의 보안 취약점 발표 사례인 OWASP(Open Web Application Security Project) Top 10을 참고하여 KISA(한국인터넷진흥원)에서 SW 보안 약점 가이드를 발표하였고 SW 보안 취약점, 약점 및 대응 방안이 구체적으로 서술되어 있음

구분	내 용
입력 데이터 검증 및 표현	XSS, SQL 인젝션을 방지하기 위해 소스코드 취약점 점검
API 이용	gets(), system.exit() 등 시스템 접근 API 오용
보안 특성	인증, 접근 제어, 기밀성, 암호화, 권한 관리, 취약한 알고리즘, 부적절 인가로 인한 취약점
시간 및 상태	프로세스 동시 수행 시, system call 등을 동시 수행 시 잘못된 권한 위임 가능성
에러 처리	에러 처리가 부적절하거나 에러에 정보가 과도하게 많이 포함된 경우
코드 품질	복잡한 소스코드가 가독성과 유지 보수성을 저하함.
캡슐화	중요 데이터의 불충분한 캡슐화로 악의적 접근 가능

인터페이스 보안

❖ 인터페이스 구현

✓ 인터페이스 보안 기능 적용

□ 데이터베이스 영역

- ✓ 데이터베이스의 기밀성을 유지하기 위해 중요 민감 데이터는 암호화하는 기법을 사용
- ✓ 데이터베이스 암호화 알고리즘은 대칭 키, 해시, 비대칭 키 알고리즘을 사용

구분	내 용
대칭 키 암호 알고리즘	ARIA 128/192/256, SEED
해시 알고리즘	SHA -256/384/512, HAS-160
비대칭 키 알고리즘	RSA, ECDSA

인터페이스 보안

❖ 인터페이스 구현

✓ 인터페이스 보안 기능 적용

□ 데이터베이스 영역

- 데이터베이스 암호화 기법으로는 애플리케이션에서 암호화를 수행하는 API 방식과 데이터베이스에서 암호화를 수행하는 Plug-in 방식, API 방식, 두가지 방식을 혼합한 Hybrid 방식이 있음

구분	API 방식	Filter(Plug-in) 방식	Hybrid 방식
개념	APP 레벨에서 암호 모듈(API)을 적용하는 APP 수정 방식	DB 레벨의 확장성 프러 시저 기능을 이용, DBMS 에 plug-in 또는 snap-in 모듈로 동작하는 방식	API 방식과 Filter 방식을 결합하거나, Filter 방식에 추가적으로 SQL문에 대한 최적화를 대항해 주는 어플라이언스를 제공하는 방식
암호화/보안 방식	별도의 AP 개발/통합	DB 내 설치/연동	어플라이언스/DB 내 설치
서버 성능 부하	APP 서버에 암호/복호화, 정책 관리, 키 관리 부하 발생	DB 서버에 암호/복호화, 정책 관리, 키 관리 부하 발생	DB와 어플라이언스에서 부하 분산
시스템 통합 용이성	APP 개발 통합 기간 필요	APP 변경 불필요	APP 변경 불필요
관리 편의성	APP 변경 및 암호화 필드 변경에 따른 유지 보수 필요	관리자용 GUI 이용, 다수 DB 통합 관리 기능 편의성 높음.	관리자용 GUI 이용, 다수 DB 통합 관리 기능 편의성 높음.

인터페이스 보안

❖ 인터페이스 구현

✓ 데이터 무결성 검사 도구

- ❑ 인터페이스 보안 취약점을 분석하는데 사용하는 도구
- ❑ 시스템 파일의 변경 유무를 확인하고 파일이 변경되었을 경우 관리자에게 알려줌
- ❑ Tripwire, AIDE, Samhain, Claymore, Slipwire, Fcheck 등



인터페이스 기능 구현 검증

❖ 인터페이스 구현 검증 도구

- ✓ 인터페이스 구현을 검증하기 위해서는 인터페이스 단위 기능 및 시나리오에 기반한 통합 테스트가 필요한데 테스트 자동화 도구를 이용하여 단위 및 통합 테스트의 효율성을 높일 수 있음

도구	설명
xUnit	java(Junit), C++(Cppunit), .Net(Nunit) 등 다양한 언어를 지원하는 단위 테스트 프레임워크
STAF	서비스 호출, 컴포넌트 재사용 등 다양한 환경을 지원하는 테스트 프레임워크
FitNesse	웹 기반 테스트 케이스 설계/실행/결과 확인 등을 지원하는 테스트 프레임워크
NTAF	Naver 테스트 자동화 프레임워크이며, STAF와 FitNesse를 통합
Selenium	다양한 브라우저 지원 및 개발언어를 지원하는 웹 애플리케이션 테스트 프레임워크
watir	Ruby 기반 웹 애플리케이션 테스트 프레임워크

인터페이스 기능 구현 검증

❖ 인터페이스 구현 감시 도구

- ✓ 인터페이스 동적 상태는 APM(Application Performance Management/Monitoring, 애플리케이션 성능 관리)을 사용하여 감시
- ✓ APM을 통해 데이터베이스와 웹 애플리케이션의 트랜잭션, 변수값, 호출 함수, 로그 및 시스템 부하 등 종합적인 정보를 조회하고 분석
- ✓ 리소스 방식과 End-To-End 방식이 있음
- ✓ APM 도구
 - ❑ 리소스 방식: Nagios, Zabbix, Cacti 등
 - ❑ End-To-End 방식: VisualVM, Scouter, Jennifer

